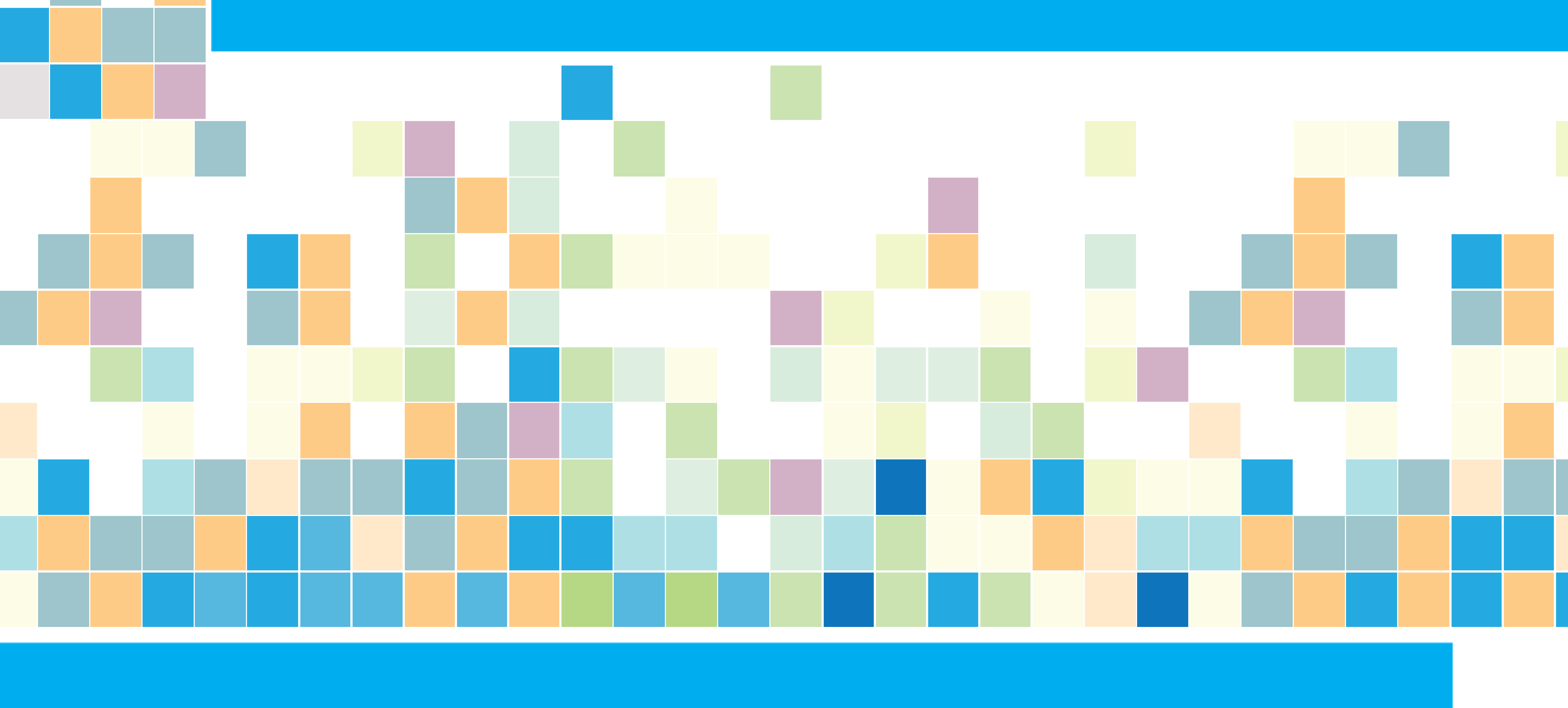
21世纪高等学校计算机教育实用规划教材

Android程序设计与应用

丁伟雄　宋晓光　李伟平　编著



清华大学出版社

21世纪高等学校计算机教育实用规划教材

Android程序设计与应用

丁伟雄　宋晓光　李伟平　编著

清华大学出版社
北京

内容简介

本书是以 Android 目前最新的版本 4.4.2 为平台编写的，书中内容全面、详细，实例丰富、实用性强，书中对每一个知识点都做了介绍，并给出一个相应的实例进行说明，使读者更快、更好地掌握 Android。本书主要是从 Android 开发最简单的内容开始，逐步深入，最后结合项目的开发进行详细讲解。

本书共 10 章，先介绍 Android 软件的基础知识、组成、布局、控件、菜单与对话框等基本内容，让读者熟悉并掌握 Android 软件，接着介绍 Android 图形、动画、存储、手机通信、手机服务、手机多媒体等内容，让读者熟练地使用 Android 进行手机功能的开发。

本书适合不同层次的读者阅读，特别适合程序开发员作为 Android 开发的参考书。

图书在版编目(CIP)数据

Android 程序设计与应用/丁伟雄，宋晓光，李伟平编著. --北京：清华大学出版社，2014
21 世纪高等学校计算机教育实用规划教材
ISBN 978-7-302-37369-8

Ⅰ. ①A…　Ⅱ. ①丁…②宋…③李…　Ⅲ. ①移动终端－应用程序－程序设计　Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2014)第 163182 号

责任编辑：魏江江　王冰飞
封面设计：
责任校对：时翠兰
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　**印　张**：31.25　**字　　数**：762 千字
版　　次：2014 年 11 月第 1 版　**印　　次**：2014 年 11 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：061037-01

出版说明

随着我国高等教育规模的扩大以及产业结构调整的进一步完善，社会对高层次应用型人才的需求将更加迫切。各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，合理调整和配置教育资源，在改革和改造传统学科专业的基础上，加强工程型和应用型学科专业建设，积极设置主要面向地方支柱产业、高新技术产业、服务业的工程型和应用型学科专业，积极为地方经济建设输送各类应用型人才。各高校加大了使用信息科学等现代科学技术提升、改造传统学科专业的力度，从而实现传统学科专业向工程型和应用型学科专业的发展与转变。在发挥传统学科专业师资力量强、办学经验丰富、教学资源充裕等优势的同时，不断更新教学内容、改革课程体系，使工程型和应用型学科专业教育与经济建设相适应。计算机课程教学在从传统学科向工程型和应用型学科转变中起着至关重要的作用，工程型和应用型学科专业中的计算机课程设置、内容体系和教学手段及方法等也具有不同于传统学科的鲜明特点。

为了配合高校工程型和应用型学科专业的建设和发展，急需出版一批内容新、体系新、方法新、手段新的高水平计算机课程教材。目前，工程型和应用型学科专业计算机课程教材的建设工作仍滞后于教学改革的实践，如现有的计算机教材中有不少内容陈旧(依然用传统专业计算机教材代替工程型和应用型学科专业教材)，重理论、轻实践，不能满足新的教学计划、课程设置的需要；一些课程的教材可供选择的品种太少；一些基础课的教材虽然品种较多，但低水平重复严重；有些教材内容庞杂，书越编越厚；专业课教材、教学辅助教材及教学参考书短缺，等等，都不利于学生能力的提高和素质的培养。为此，在教育部相关教学指导委员会专家的指导和建议下，清华大学出版社组织出版本系列教材，以满足工程型和应用型学科专业计算机课程教学的需要。本系列教材在规划过程中体现了如下一些基本原则和特点。

(1) 面向工程型与应用型学科专业，强调计算机在各专业中的应用。教材内容坚持基本理论适度，反映基本理论和原理的综合应用，强调实践和应用环节。

(2) 反映教学需要，促进教学发展。教材规划以新的工程型和应用型专业目录为依据。教材要适应多样化的教学需要，正确把握教学内容和课程体系的改革方向，在选择教材内容和编写体系时注意体现素质教育、创新能力与实践能力的培养，为学生知识、能力、素质协调发展创造条件。

(3) 实施精品战略，突出重点，保证质量。规划教材建设仍然把重点放在公共基础课和专业基础课的教材建设上；特别注意选择并安排一部分原来基础比较好的优秀教材或讲义修订再版，逐步形成精品教材；提倡并鼓励编写体现工程型和应用型专业教学内容和课程体系改革成果的教材。

（4）主张一纲多本，合理配套。基础课和专业基础课教材要配套，同一门课程可以有多本具有不同内容特点的教材。处理好教材统一性与多样化，基本教材与辅助教材，教学参考书，文字教材与软件教材的关系，实现教材系列资源配套。

（5）依靠专家，择优选用。在制订教材规划时要依靠各课程专家在调查研究本课程教材建设现状的基础上提出规划选题。在落实主编人选时，要引入竞争机制，通过申报、评审确定主编。书稿完成后要认真实行审稿程序，确保出书质量。

繁荣教材出版事业，提高教材质量的关键是教师。建立一支高水平的以老带新的教材编写队伍才能保证教材的编写质量和建设力度，希望有志于教材建设的教师能够加入到我们的编写队伍中来。

21世纪高等学校计算机教育实用规划教材编委会

联系人：魏江江 weijj@tup.tsinghua.edu.cn

前　　言

Android 是一个开放式手机和平板电脑的操作系统，目前的发展十分迅猛。虽然 Android 面世时间不长，但 Android 已经对传统的手机平台构成了强大的威胁。业界部分人士预测，Android 将会成为应用最广泛的手机操作系统。

Android 是 Google 于 2007 年 11 月 5 日发布的基于 Linux 平台的开源移动操作系统，Google 向全世界推广 Android 的策略是持续的，而且推广的力度很大。随着越来越多硬件厂商的加入，精心设计的程序通过各种接口到达 Android 终端设备，如手机、平板电脑、手持游戏设备、数字相框、电子书和 Google TV 电视盒等。

Android 作为一个开放、开源的移动终端平台，对于业界来讲，这意味着源代码基于 Apache 2.0 许可进行开放。Android 具有以下五大优势。

(1) 开放性：Android 平台的最大优势就是其开放性，开放的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益增加，一个崭新的平台也将很快走向成熟。

开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，最大的受益正是丰富的软件资源。开放的平台也会带来更大的竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

(2) 挣脱运营商的束缚：在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。随着 EDGE、HSDPA 这些 2G 至 3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈。

(3) 丰富的硬件选择：这与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪、功能特色各具的多种产品。功能上的差异和特色，不会影响到数据同步，甚至是软件的兼容。

(4) 不受任何限制的开发商：Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的限制，可想而知，会有多少新颖别致的软件产生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如何控制是留给 Android 的难题之一。

(5) 无缝结合的 Google 应用：在互联网上 Google 已经有了十几年历史，从搜索巨人到全面的互联网渗透，Google 服务(如地图、邮件、搜索等)已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝地结合这些优秀的 Google 服务。

在 Android 推出之前，移动开发领域的发展一直处于不温不火的局面，Android 的推出为移动互联网开发领域吹进了一股清新的风。它的精巧体系架构以及完全开放的特性也吸引了无数的开发人员。Android 作为一个优秀的移动操作平台，其程序开发的学习很难，最大的困难就是相关资源的缺乏。Google 提供的主要学习资料就是 Android SDK 文档。SDK 文档对于开发人员了解 Android 程序设计有很大的帮助，但并没有系统地讲解

Android 程序设计的相关技术。针对这些问题,作者精心编写了本书。

本书的编写具有以下几大优点。

- 内容全:对于刚接触 Android 的人员,本书首先对 Android 系统的历史以及架构做了详细的介绍。
- 实例多:对于 Android 系统中的每一个知识点,不管是一个简单的文本框还是复杂的控件,本书都会给出一个相应的实例做说明,这样便于读者对知识点进行理解和掌握。
- 实用性强:本书采用 Android 应用程序常用的知识点,并结合实例讲解,让读者在实际应用中能够快速上手,同时也方便读者对程序进一步扩展。
- 通俗易懂:本书条理清晰、文字简洁,每讲解一些基本概念都结合实例进行说明,做到理论与实践相结合,从而让读者快速理解与掌握 Android 的相关应用。
- 图文并茂:针对没有接触过 Android 的读者,本书对相关概念一般会插入对应的图片做说明,同时对每一个知识点实例的运行效果给出相应的运行效果图,这样对读者掌握这一知识点起到了很大的作用。

应业界需要,作者编写了本书,本书的编写平台是目前 Android 的最新版本 4.4.2。本书共分为 10 章,其主要内容如下。

第 1 章:介绍 Android 入门知识,主要包括 Android 概述、Android 环境搭建、Android 应用项目组成等内容。

第 2 章:介绍 Android 基本组件,主要包括 Android 生命周期、资源的管理与使用、Activity(活动)、Intent(意图)等内容。

第 3 章:介绍 Android 布局,主要包括 UI 界面、View 对象、布局管理器等内容。

第 4 章:介绍 Android 基本控件,主要包括文本类控件、按钮类控件、列表类控件等内容。

第 5 章:介绍 Android 菜单与对话框,主要包括 Android 菜单、Android 对话框、Android 消息提示框等内容。

第 6 章:介绍 Android 图形与动画,主要包括 Android 图形、Path 绘图、Android 动画等内容。

第 7 章:介绍 Android 持久化存储,主要包括文件结构、数据存储方式、Android 的各种存储等内容。

第 8 章:介绍 Android 手机通信与服务,主要包括 RPC 通信、TCP 通信、UDP 通信以及手机服务等内容。

第 9 章:介绍 Android 手机自动控制服务,主要包括查看手机信息、查看 SIM 信息、闹钟设置等内容。

第 10 章:介绍 Android 的多媒体功能,主要包括音频播放、录制多媒体、视频播放以及感应器等内容。

本书主要由丁伟雄、宋晓光和李伟平编写,此外参加编写的人员还有刘超、邓俊辉、梁朗星、李旭波、张棣华、刘泳、邓耀隆、何正风和周品。

本书适合不同层次的读者阅读,特别适合程序开发员作为 Android 开发的参考书。由于作者的水平有限,加之时间仓促,书中难免会存在不足之处,敬请广大读者批评指正。

编者

2014 年 10 月

目　　录

第 1 章　Android 入门知识

Android 是一种基于 Linux 的自由及开放源代码的操作系统，主要用于移动设备，例如智能手机和平板电脑，由 Google 公司和开放手机联盟领导及开发。Android 操作系统最初由 Andy Rubin 开发，主要支持手机。2005 年 8 月被 Google 公司收购。2007 年 11 月，Google 与 84 家硬件制造商、软件开发商及电信营运商组建开放手机联盟共同研发改良 Android 系统。随后，Google 以 Apache 开源许可证的授权方式发布了 Android 的源代码。第一部 Android 智能手机发布于 2008 年 10 月。Android 逐渐扩展到平板电脑及其他领域，例如电视、数码相机、游戏机等。2011 年第一季度，Android 在全球的市场份额首次超过塞班系统，跃居全球第一。2012 年 11 月数据显示，Android 占据全球智能手机操作系统市场 76％的份额，中国市场占有率为 90％。2013 年 9 月 24 日，谷歌开发的操作系统 Android 迎来了 5 岁生日，全世界采用这款系统的设备数量已经达到 10 亿台。

1.1　Android 概述

Android 一词的本义指“机器人”，同时也是 Google 公司于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称，该平台由操作系统、中间件、用户界面和应用软件组成。

1.1.1　Android 平台特性

Android 号称是首个为移动终端打造的真正开放和完整的移动平台，是安全开源免费的操作系统，任何人都可以获得和使用 Android 系统。Google 公司还提供了 Android SDK，包括进行 Android 应用开发所必需的工具和 API 接口。

Android 操作系统具有以下特性：

- 灵活的应用程序框架，可以随意重复使用或者替换手机的组件。
- 提供了专为移动设备优化的虚拟机——Dalvik 虚拟机。
- 拥有内部集成的浏览器——基于开源的 WebKit 引擎。
- 提供针对手机优化的图形库，包括定制的 2D 图形库和基于 OpenGL ES 1.0 的 3D 图形库。
- 使用集成了轻量级数据库管理系统 SQLite 作为结构化的数据存储。
- 娱乐功能丰富，支持多种媒体格式。
- 支持多种移动电话技术，例如 GSM、WCDM 等。
- 支持 USB、蓝牙、WiFi 等多种数据传输。

- 支持摄像头、GPS、光线传感器、加速传感器、温度传感器等多种传感器。
- 提供了丰富的开发工具，包括设备模拟器、调试工具、内存及性能分析图表和 Eclipse 集成开发环境插件等。

目前，Android 系统不仅应用于智能手机，还在平板电脑领域急速扩张。2011 年初数据显示，正式上市仅两年多的 Android 系统已经超越称霸 10 年的 Symbian（塞班）系统，并跃居全球最受欢迎的智能手机平台。随着 Android 越来越火，不少 Android 开发人员难免会被问到这样的问题，就是这个平台有什么优势，当然它也有一些不足。

Android 系统具有以下优势。

(1) 开放性：Android 平台首先就是其开放性，开放的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益增加，一个崭新的平台也将很快走向成熟。

开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，最大的受益正是丰富的软件资源。开放的平台也会带来更大的竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

(2) 挣脱运营商的束缚：在过去很长的一段时间内，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。自从 iPhone 上市后，用户可以更加方便地连接网络，运营商的制约减少。随着 EDGE、HSDPA 这些 2G 至 3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈。

互联网巨头 Google 推动的 Android 终端天生就有网络特色，让用户离互联网更近。

(3) 丰富的硬件选择：这一点与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪、功能特色各具的多种产品。功能上的差异和特色，不会影响到数据同步，甚至是软件的兼容，好比你从诺基亚 Symbian 风格手机一下改用苹果 iPhone，同时还可以将 Symbian 中优秀的软件带到 iPhone 上使用，联系人等资料更是可以方便地转移。

(4) 不受任何限制的开发商：Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的限制，可想而知，会有多少新颖别致的软件产生。但也有其两面性，血腥、暴力、情色等方面的程序和游戏如何控制是留给 Android 的难题之一。

(5) 无缝结合的 Google 应用：如今叱咤互联网的 Google 已经走过十几年的历史，从搜索巨人到全面的互联网渗透，Google 服务（如地图、邮件、搜索等）已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝地结合这些优秀的 Google 服务。

当然，“金无足赤”，相对于其他一些智能手机操作系统而言，由于进入市场的时间不长，作为后起之秀的 Android 在现阶段还存在着以下不足。

(1) 安全和隐私：由于手机与互联网紧密联系，个人隐私很难得到保护。除了上网过程中经意或不经意地留下个人信息外，Google 时时刻刻洞察着一切，因此，互联网的深入将会带来新的隐私危机。

(2) 运营商仍然能够影响到 Android 手机：在国内市场，不少用户对购得的移动定制机不满，感觉所购的手机被人加了广告。这样的情况在国外市场同样出现。Android 手机的另一发售运营商 Sprint 就在其机型中内置了其手机商店程序。

(3) 同类机型用户减少：在不少手机论坛都会有针对某一型号的子论坛，对一款手机

的使用进行交流，并分享软件资源。而对于 Android 平台手机，由于厂商较多、产品类型多样，这样使用同一款机型的用户越来越少，缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，很少有专门针对某个型号的山寨机的讨论和群组，除了那些功能异常抢眼、颇受追捧的机型以外。

(4) 过分依赖开发商，缺少标准配置：在使用 PC 端的 Windows XP 系统的时候，都会内置微软 Windows Media Player 这样一个播放器程序，用户可以选择更多的播放器，例如 RealPlayer 或暴风影音等，但开始时使用默认的程序同样可以应付多样的需要。在 Android 平台中，由于其开放性，软件更多依赖第三方厂商，例如 Android 系统的 SDK 中就没有内置音乐播放器，全部依赖第三方开发，缺少了产品的统一性。

1.1.2 Android 平台架构

Android 系统是以 Linux 系统为基础的，Google 公司将其按功能特性划分为 4 层，自下而上分别是 Linux 内核、中间件、应用程序框架和应用程序，如图 1-1 所示。

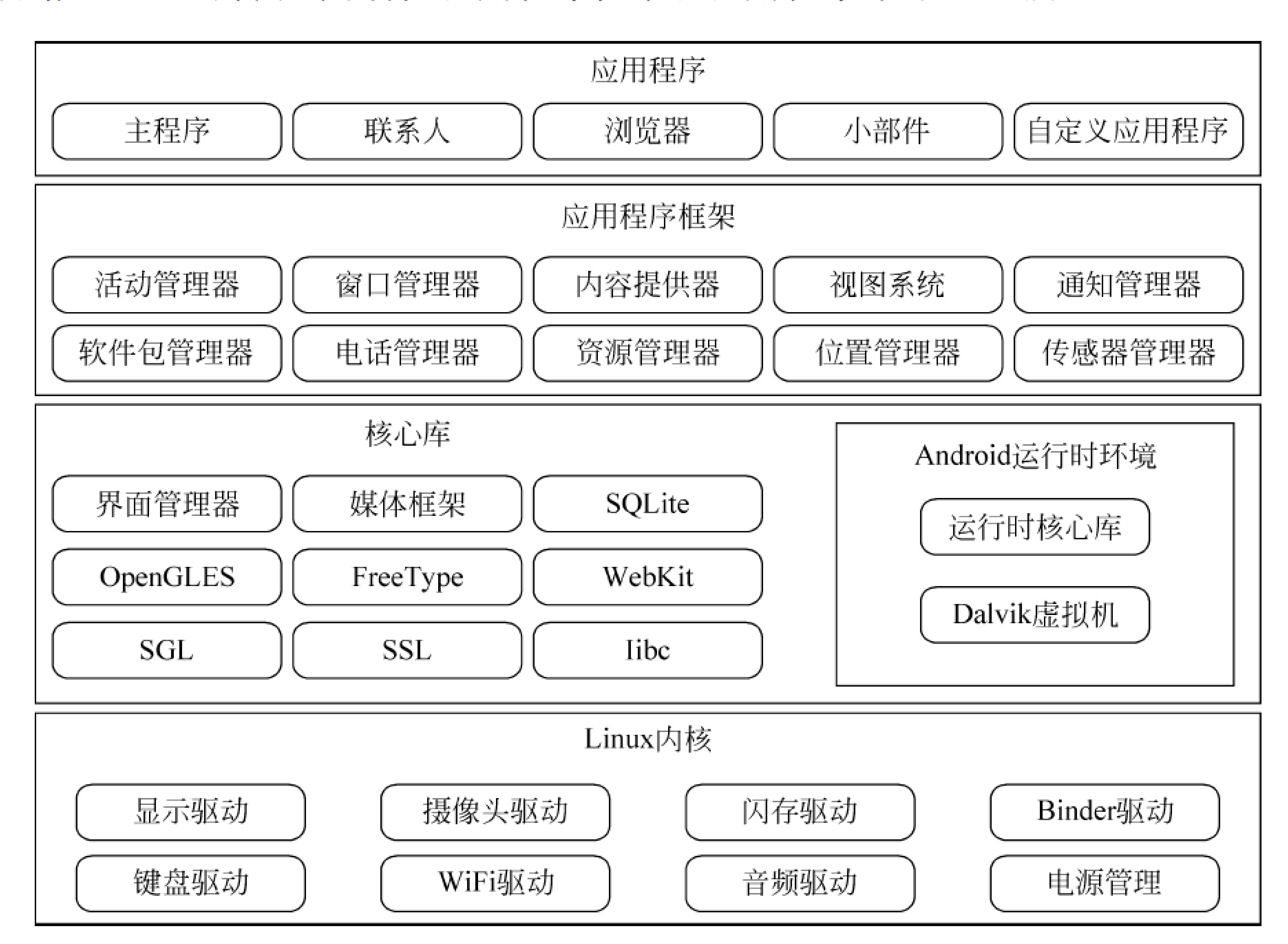


图 1-1 Android 系统框架图

1. 应用程序

Android 系统内置了一些常用的应用程序，包括 Home 视图、联系人、电话、浏览器等。这些应用程序和用户自己编写的应用程序一样，都是采用 Java 语言编写的。而且，用户可以根据需要增加自己的应用程序，或者替换系统自带的应用程序。

2. 应用程序框架

应用程序框架提供了程序开发人员的接口，这是与 Android 程序员直接相关的部分。开发者可以用它开发应用，其中包括以下内容。

- 丰富且可扩展的视图(Views)：可以用来构建应用程序，包括列表(Lists)、网格

(Grids)、文本框(Text Boxes)、按钮(Buttons),甚至可嵌入的 Web 浏览器。

- 内容提供器(Content Providers):使得应用程序可以访问另一个应用程序的数据(例如联系人数据库),或者共享它们自己的数据。
- 资源管理器(Resource Manager):提供非代码资源的访问,例如本地字符串、图形、布局文件(layoutfiles)。
- 通知管理器(Notification Manager):使得应用程序可以在状态栏中显示自定义的提示信息。
- 活动管理器(Activity Manager):用来管理应用程序生命周期,并提供常用的导航回退功能。

3. 中间件

中间件包括两个部分,即核心库(libraries)和 Android 运行时环境(Android runtime)。

1) 核心库

核心库中主要包括一些 C/C++核心库,以方便开发者进行应用的开发。

- 系统 C 库(libc):专门为基于 Embedded Linux 的设备定制的。
- 媒体库:支持多种常用的音频、视频格式回放和录制,同时支持静态图像文件。编码格式包括 MPEG4、H.264、MP3、AAC、AMR、JPG、PNG。
- Surface Manager:对显示子系统进行管理,并且为多个应用程序提供了 2D 和 3D 图层的无缝融合。
- WebKit/LibWebCore:Web 浏览引擎,支持 Android 浏览器和一个可嵌入的 Web 视图。
- SGL:底层的 2D 图形引擎。
- 3D libraries:基于 OpenGL ES 1.0 APIs 实现的 3D 引擎。
- FreeType:位图(bitmap)和矢量(vector)字体显示。
- SQLite:轻型关系型数据库引擎。

2) Android 运行时环境

Android 运行时环境主要包括以下内容。

- 运行时核心库:提供了 Java 库的大多数功能。
- Dalvik 虚拟机:依赖于 Linux 内核的一些功能,例如线程机制和底层内存管理机制。同时,虚拟机是基于寄存器的,Dalvik 采用简练、高效的 byte code 格式运行,它能够在低耗资和没有应用相互干扰的情况下并行执行多个应用,每一个 Android 应用程序都在它自己的进程中运行,都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 虚拟机中的可执行文件的扩展名为.dex,该格式的文件针对小内存使用做了优化。所有的类都经由 Java 编译器编译,然后通过 SDK 中的 dx 工具转换成.dex 格式,由虚拟机执行。

4. Linux 内核

Android 平台运行在 Linux 2.6 之上,其 Linux 内核部分相当于手机硬件层和软件层之间的一个抽象层。Android 的内核提供了显示驱动、摄像头驱动、闪存驱动、键盘驱动、WiFi 驱动、音频驱动和电源管理等多项功能。此外,Android 为了让 Android 程序可以用于商业目的,将 Linux 系统中受 GNU 协议约束的部分进行了取代。

1.1.3 Android 市场

Android 市场是 Google 公司为 Android 平台提供的在线应用商店，Android 平台用户可以在该市场中浏览、下载和购买第三方人员开发的应用程序。

对于开发人员，有两种获利方式：一种方式是销售软件，开发人员可以获得该应用销售额的 70%，其余 30%作为其他费用；另一种方式是加广告，即将自己的软件定为免费软件，通过广告链接增加点击率获得。

1.1.4 Android 应用组件

Android 开发有四大组件，其中，活动(Activity)用于表现功能；服务(Service)为后台运行服务，不提供界面呈现；广播接收器(Broadcast Receiver)用于接收广播；内容提供商(Content Provider)支持在多个应用中存储和读取数据，相当于数据库。

1. 活动

在 Android 中，Activity 是所有程序的根本，所有程序的流程都运行在 Activity 之中，Activity 可以算是开发者遇到的最频繁，也是 Android 当中最基本的模块之一。在 Android 的程序当中，Activity 一般代表手机屏幕的一屏。如果把手机比作一个浏览器，那么 Activity 就相当于一个网页。在 Activity 当中可以添加一些 Button、Check Box 等控件，可以看到 Activity 概念和网页的概念相当类似。

通常，一个 Android 应用是由多个 Activity 组成的。这多个 Activity 之间可以进行相互跳转，例如，按下一个 Button 按钮后，可能会跳转到其他的 Activity。和网页跳转稍微有些不一样的是，Activity 之间的跳转有可能返回值，例如，从 Activity A 跳转到 Activity B，那么当 Activity B 运行结束的时候，有可能会给 Activity A 一个返回值。这样做在很多时候是相当方便的。

Android 的 4 种 Activity 加载模式流程如图 1-2 所示。

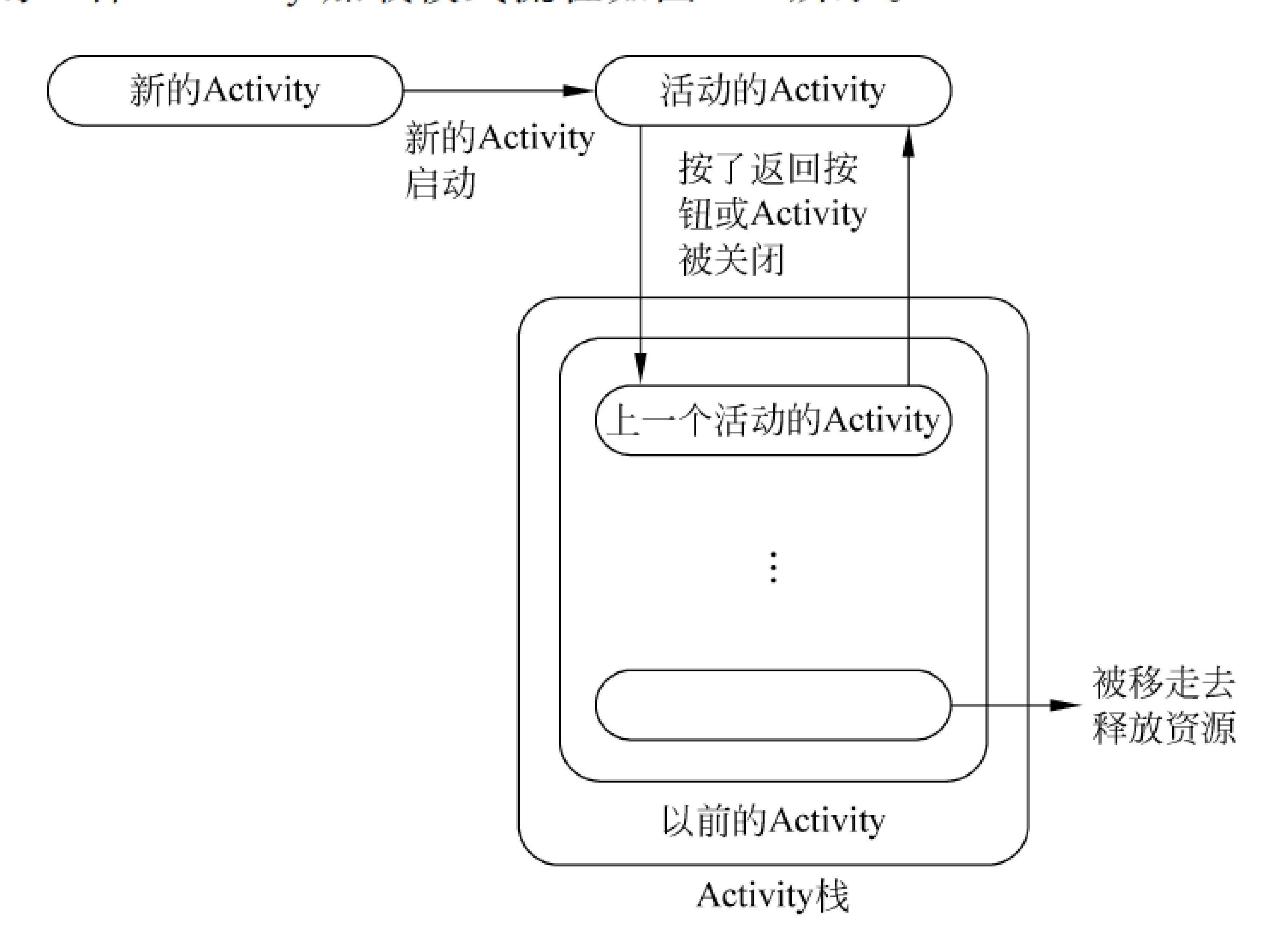


图 1-2　Android 的 4 种 Activity 加载模式流程图

当打开一个新的屏幕时，之前一个屏幕会被置为暂停状态，并且压入历史堆栈中。用户可以通过回退操作返回到以前打开过的屏幕，可以选择性地移除一些没有必要保留的屏幕，因为 Android 会把每个应用的开始到当前的每个屏幕都保存在堆栈中。

2. 服务

Service 是 Android 系统中的一种组件，它跟 Activity 的级别差不多，但是它不能自己运行，只能后台运行，并且可以和其他组件进行交互。Service 是没有界面的长生命周期的代码。Service 是一种程序，它可以运行很长时间，但是它却没有用户界面。这么说有点枯燥，下面来看一个例子。打开一个音乐播放器的程序，如果想上网了，那么打开 Android 浏览器，这个时候虽然已经进入了浏览器这个程序，但是歌曲的播放并没有停止，而是在后台继续一首接着一首播放。其实这个播放就是由播放音乐的 Service 进行控制的。当然这个播放音乐的 Service 也可以停止，例如，当播放列表里的歌曲都结束时，或者用户按下了停止音乐播放的快捷键等。Service 可以在多场合的应用中使用，比如播放多媒体的时候用户启动了其他 Activity，这个时候程序要在后台继续播放；比如检测 SD 卡上文件的变化；再或者在后台记录地理信息位置的改变等，而服务则藏在后台。

开启 Service 有下面两种方式。

(1) Context. startService()：Service 会经历 onCreate→onStart(如果 Service 还没有运行，则 Android 先调用 onCreate()，然后调用 onStart()；如果 Service 已经运行，则只调用 onStart()，所以一个 Service 的 onStart()方法可能会重复调用多次)；StopService 的时候直接 onDestroy，如果是调用者自己直接退出而没有调用 StopService，Service 会一直在后台运行。该 Service 的调用者在启动起来后可以通过 StopService 关闭 Service。注意，多次调用 Context. startService()不会嵌套(即使会有相应的 onStart()方法被调用)，所以无论同一个服务被启动了多少次，一旦调用 Context. stopService()或者 StopSelf()，它都会被停止。补充说明：传递给 StartService 的 Intent 对象会传递给 onStart()方法。调用顺序为 onCreate →onStart(可多次调用) →onDestroy。

(2) Context. bindService()：Service 会经历 onCreate()→onBind()，onBind()将返回给客户端一个 IBind 接口实例，IBind 允许客户端回调服务的方法，比如得到 Service 运行的状态或其他操作。这个时候调用者(Context，例如 Activity)会和 Service 绑定在一起，Context 退出了，Srevice 就会调用 onUnbind→onDestroyed 相应退出，所谓绑定在一起即为“共存亡”。

3. 广播接收器

在 Android 中，Broadcast 是一种广泛运用在应用程序之间传输信息的机制。而 BroadcastReceiver 是对发送出来的 Broadcast 进行过滤接受并响应的一类组件，可以使用 BroadcastReceiver 让应用对一个外部的事件做出响应。这是非常有意思的，例如，当电话呼入这个外部事件到来的时候，可以利用 BroadcastReceiver 进行处理。例如，当下载一个程序成功完成的时候，仍然可以利用 BroadcastReceiver 进行处理。BroadcastReceiver 不能生成 UI，也就是对于用户来说不是透明的，用户是看不到的。BroadcastReceiver 通过 NotificationManager 来通知用户这些事情发生了。BroadcastReceiver 既可以在 AndroidManifest. xml 中注册，也可以在运行时的代码中使用 Context. registerReceiver()进行注册。只要注册了，当事件来临的时候即使程序没有启动，系统也在需要的时候启动程序。各种应用还可以通过使用 Context. sendBroadcast()将自己的 Intent Broadcasts 广播

给其他应用程序。

4. 内容提供

Content Provider 是 Android 提供的第三方应用数据的访问方案。

在 Android 中，对数据的保护是很严密的，除了放在 SD 卡中的数据，一个应用所持有的数据库、文件等内容都是不允许其他直接访问的。Android 当然不会真的把每个应用都做成一座“孤岛”，它为所有应用都准备了一扇“窗”，这就是 Content Provider。应用向对外提供的数据，可以通过派生 Content Provider 类封装成一个 Content Provider，每个 Content Provider 都用一个 URI 作为独立的标识，形如 content://com.xxxxx。所有应用看着像 REST，但实际上，它比 REST 更为灵活。和 REST 类似，URI 也可以有两种类型，一种是带 ID 的，另一种是列表的，但实现者不需要按照这个模式来做，带 ID 的 URI 也可以返回列表类型的数据。

1.2 Android 环境搭建

在搭建环境前，需要了解安装开发工具所需要的硬件和软件配置条件。

1.2.1 系统需求

本节介绍使用 Android SDK 进行开发所需要的硬件和软件配置条件。对于硬件方面，要求 CPU 和内存尽量大。Android SDK 全部下载大概需要占用 4.5GB 硬盘空间。由于开发过程中需要反复重启模拟器，而每次重启都会消耗几分钟的时间（视机器配置而定），因此使用高配置的机器能节约不少时间。

支持 Android SDK 的操作系统及其要求如表 1-1 所示。

表 1-1 Android SDK 对操作系统的要求

操作系统	要求
Windows	Windows XP(32 位) Vista(32 位或 64 位) Windows 7(32 位或 64 位)
Mac OS	10.5.8 或更新(仅支持 x86)
Linux(在 Ubuntu 的 10.04 版测试)	需要 GNU C Library(glibc)2.7 或更新 在 Ubuntu 系统上，需要 8.04 版或更新 64 位版本必须支持 32 位应用程序

对于开发环境，除了常用的 Eclipse IDE 以外，还可以使用 Intelli J IDEA 进行开发。对于 Eclipse，在下载 Android SDK 时就自带了相兼容的版本。

1.2.2 安装 JDK

在 Windows 平台上搭建 Android 开发环境，首先要下载并安装与开发环境相关的软件资源，这些资源主要包括 JDK、Eclipse、Android SDK 和 Development Tools 插件（ADT 插件）。

在 Android 平台上，所有应用程序都是使用 Java 语言编写的，所以要安装 Java 开发包 JDK(Java SE Development Kit)，JDK 是 Java 开发时所必需的软件开发包。

安装 JDK 的过程比较简单，运行该程序后，根据安装提示选择安装路径，将 JDK 安装到指定的文件夹即可，其默认的安装目标为“C:\Program Files\Java\jdk1.6.0_10(jdk-6u10-rc2-bin-b32-windows-i586-p-12_sep_2008)”。

JDK 安装完毕后，要进一步设置 Java 的环境变量，即设置 bin 和 lib 文件夹的路径。其操作步骤如下(操作系统为 Windows 7 时)：

(1) 右击“计算机”，在弹出的快捷菜单中选择“属性”选项，在打开的“系统”窗口中单击“高级系统设置”链接，弹出“系统属性”对话框，如图 1-3 所示。



图 1-3 “系统属性”对话框

(2) 在“系统属性”对话框的“高级”选项卡中单击“环境变量”按钮，弹出“环境变量”对话框，如图 1-4 所示。



图 1-4 “环境变量”对话框

(3) 选中“系统变量”列表框中的 Path 变量，单击“编辑”按钮，弹出“编辑系统变量”对话框，如图 1-5 所示。

(4) 在该对话框的“变量值”文本框中添加“C:\Program Files\Java\jdk1.6.0_10\bin”，单击“确定”按钮即可完成设置，这样即设置了 bin 文件夹的路径。

(5) 在“环境变量”对话框的“系统变量”列表框中单击“新建”按钮，弹出“新建系统变量”对话框，如图 1-6 所示。

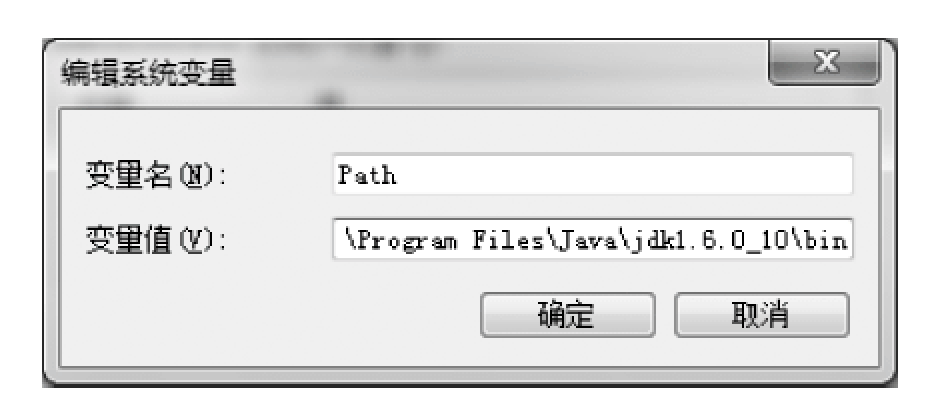


图 1-5 “编辑系统变量”对话框

图 1-6 “新建系统变量”对话框

(6) 在图 1-6 中的“变量名”文本框中输入 classpath，在“变量值”文本框中输入“C:\Program Files\Java\jdk1.6.0_10\lib”，即可设置 lib 文件夹的路径。

完成以上操作后，一个典型的 Java 开发环境便设置好了。在正式开始下一步前应先验证 Java 开发环境的设置是否成功。

在 Windows 7 系统中单击“开始”按钮，选择“附件”下的“运行”命令，在“运行”对话框中输入“cmd”并回车，即可打开 CMD 窗口，在该窗口中输入“java-version”，则可显示所安装的 Java 版本信息，如图 1-7 所示。



图 1-7 JDK 安装成功界面

1.2.3 安装 Eclipse

安装并设置好 JDK 后，即可安装 Eclipse 了。Eclipse 是一个非常强大的集成开发环境，可以支持 Java、C、C++ 等多种语言。由于 Android 是使用 Java 开发的，因此需要下载 Java 版本的 Eclipse 集成开发环境。

在下载 Android SDK 时，其压缩包中已经包含了相兼容的最新版本 Eclipse(中文版)，解压到硬盘上的某个目录即可。Eclipse 集成开发环境无须安装，进入解压后的目录双击可执行文件 eclipse.exe，Eclipse 能自动找到用户前面安装的 JDK 路径，如图 1-8 所示。



图 1-8 Eclipse 启动界面

启动 Eclipse 的开发环境界面,将会看到选择工作空间的提示,如图 1-9 所示。

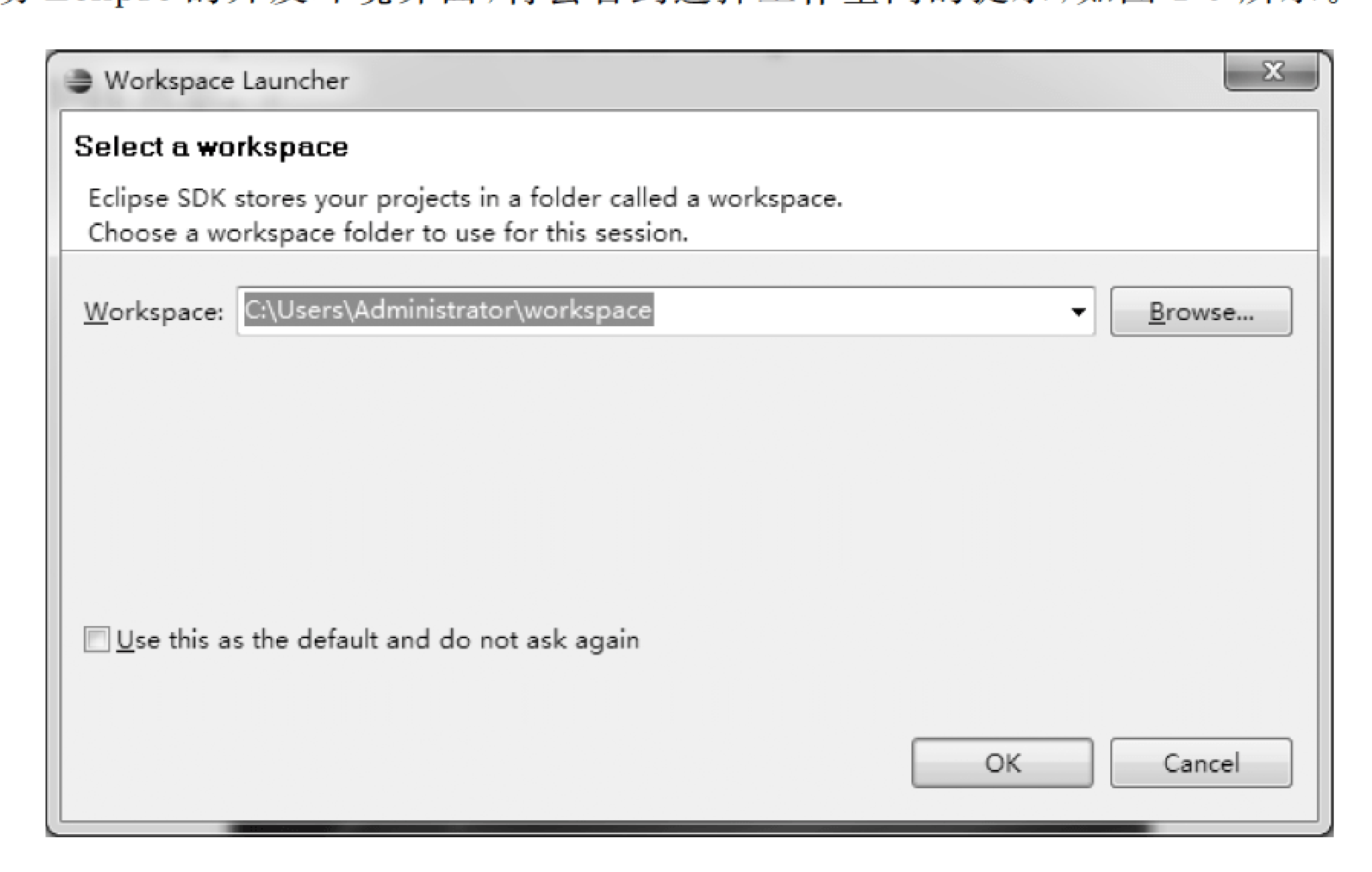


图 1-9　选择工作空间

单击图 1-9 中的 OK 按钮,即完成 Eclipse 的安装,系统进入 Eclipse 初始欢迎界面,如图 1-10 所示。接着单击图 1-10 左上角的"欢迎"按钮,即可进入 Eclipse 的开发环境界面,如图 1-11 所示。



图 1-10　Eclipse 初始欢迎界面

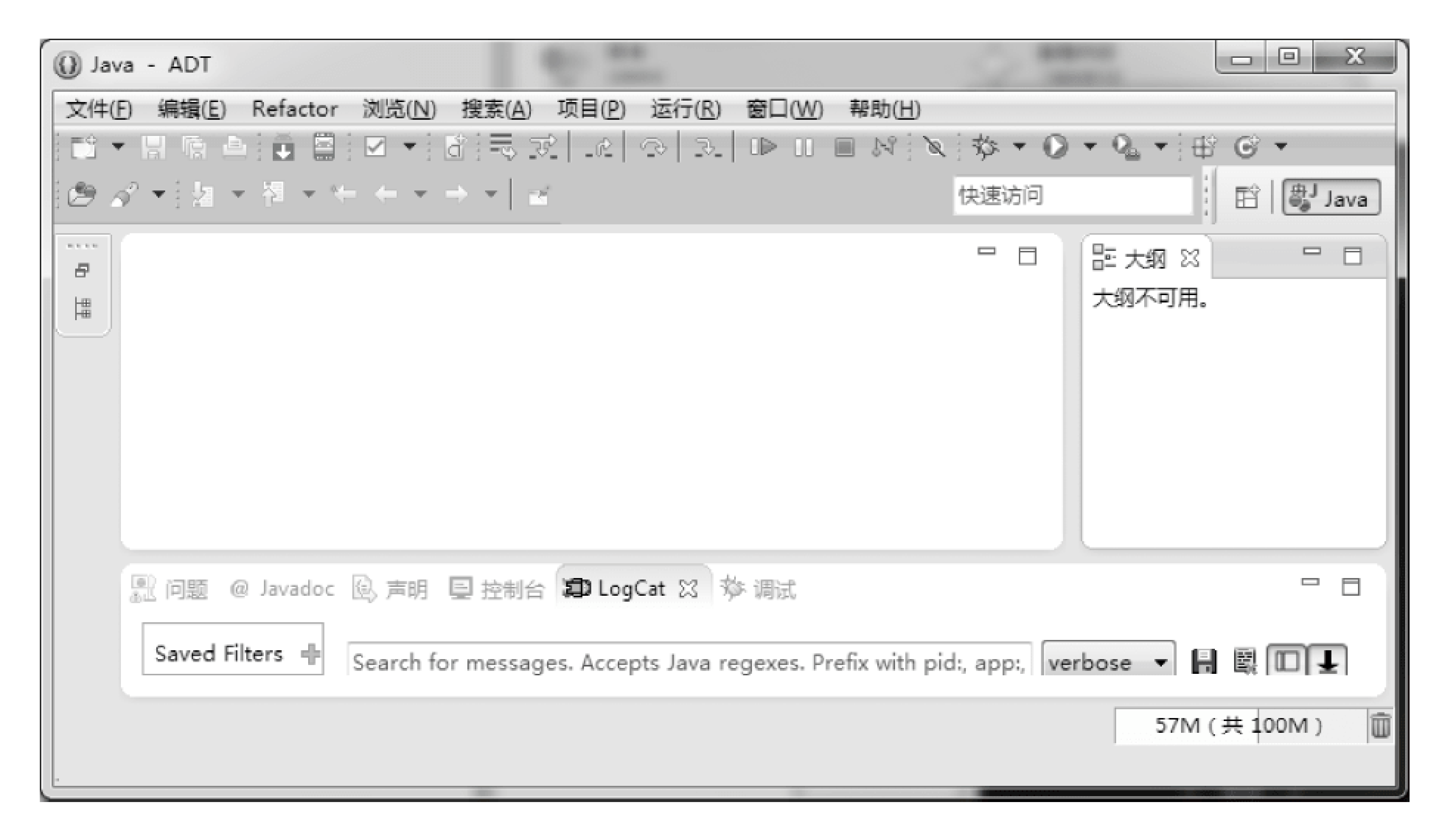


图 1-11　Eclipse 的开发环境界面

1.2.4　安装 Android SDK

将下载的 Android SDK 开发包解压到硬盘上的某个目录，该目录在后面配置 Android 开发工具 ADT 和使用 SDK 工具时都会用到。

解压后的文件夹中有以下几个重要的文件和文件夹。

- add-ons：用来保存插件工具，目前为空。
- platforms：用来保存不同版本的 SDK 数据包，目前为空。
- Tools：包含了 Android 的 SDK 工具。
- SDK Manager.exe：SDK 管理工具，可以用来更新 SDK 数据包、管理 Android 模拟器等。
- SDK Readme.txt：Android SDK 的说明文件。

Android SDK 与 Eclipse 集成开发环境一样，不需要经过真正的安装过程，相当于解压之后就可以运行。读者在第一次运行 SDK Manager 时需要下载 Android 各个版本的 SDK 数据包，操作步骤如下：

(1) 双击“SDK Manager.exe”执行文件，程序将自动检测是否有更新的 SDK 数据包可下载，如图 1-12 所示。

(2) 对于所要更新的内容，如果要尝试一下 Android 4.4.2，那么只选择“Android 4.4.2 (API 19)”，然后单击 Install packages 按钮安装就可以了。如果要在此 SDK 上开发应用程序和游戏应用，那么需要接受并遵守所有许可内容(Accept All)，并单击 Install packages 按钮。

(3) 将 SDK Tools 目录的完整路径设置到系统变量 Path 中，这样便于在后面调用 Android 命令时无须输入全部的路径。设置系统变量 Path 的方法与设置 JDK 的环境变量值的操作一样，在“编辑系统变量”对话框的“变量值”文本框中添加“;D:\Andoridtool\Android_SDK_windows\sdk\sdk\tools;”即可，如图 1-13 所示。

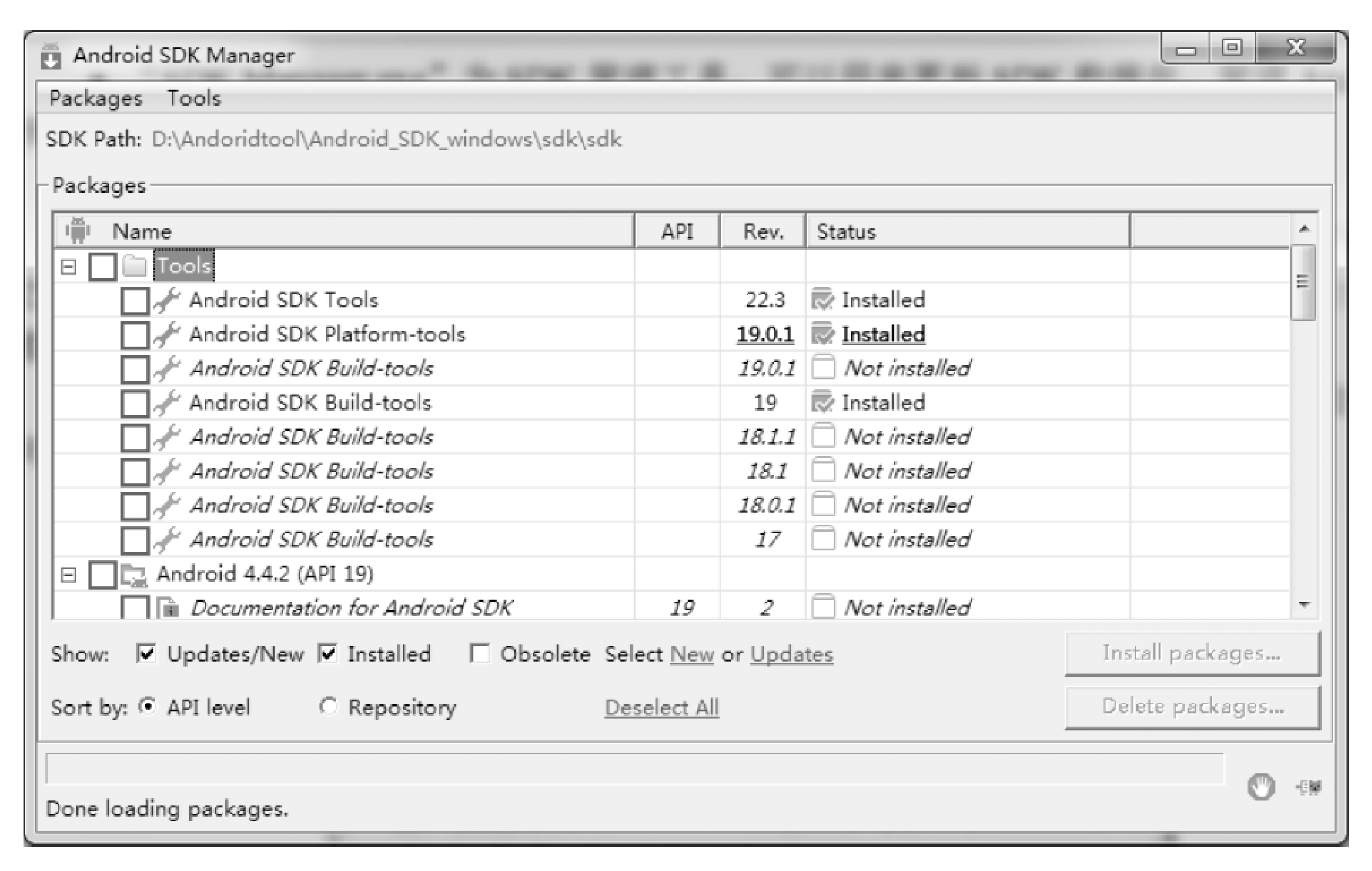


图 1-12　运行 SDK Manager. exe 执行文件



图 1-13　设置系统变量

（4）检查 Android SDK 是否安装成功，是否能够正常运行。在 Windows 7 系统中单击“开始”按钮，选择“附件”下的“运行”命令，在“运行”对话框中输入 cmd 并回车，即可打开 CMD 窗口，在窗口中输入“android-h”，则可显示所安装的 Android SDK 的信息，如图 1-14 所示。

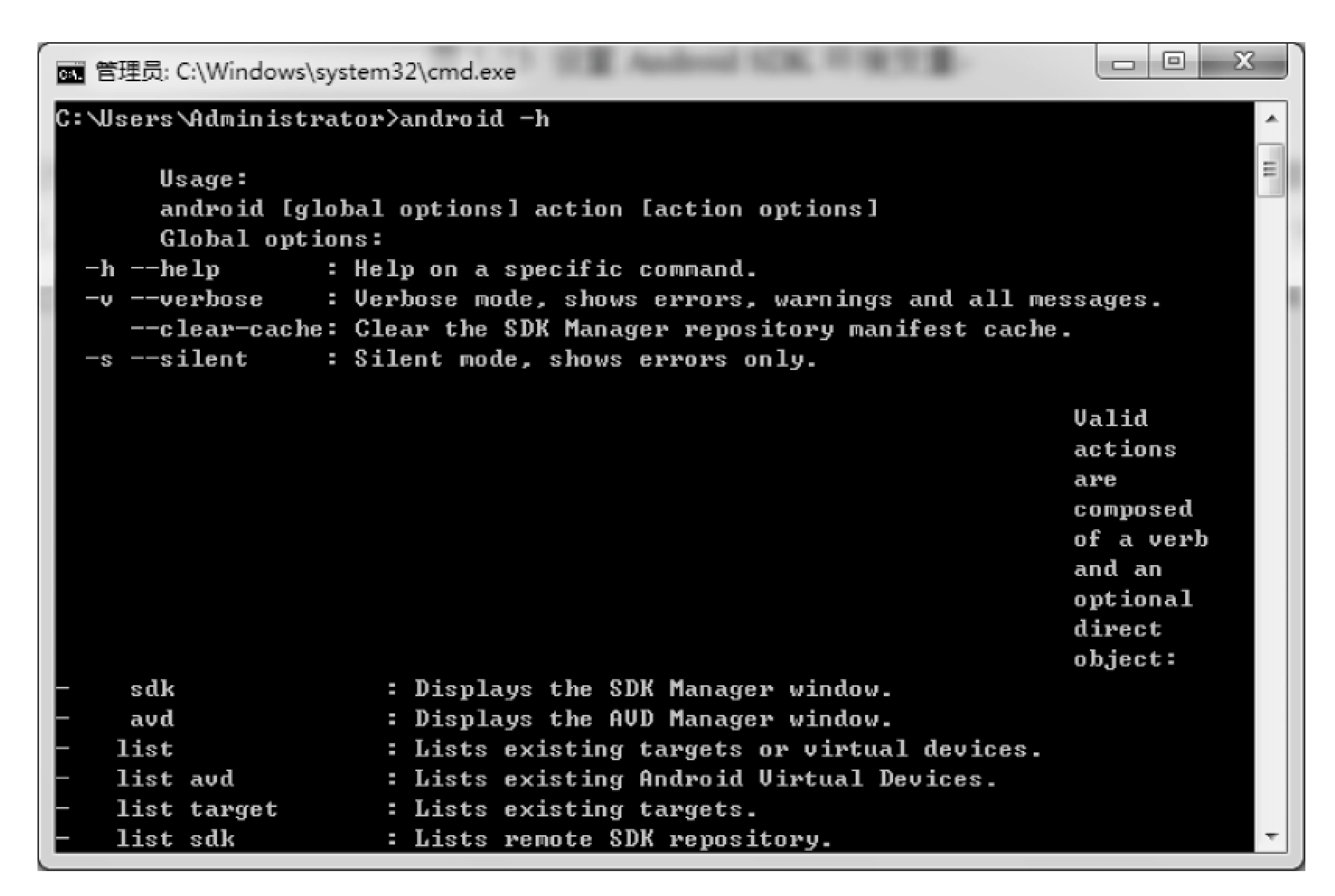


图 1-14　Android SDK 安装成功信息

1.2.5 安装 ADT 插件

Android 为 Eclipse 定制了一个插件，即 Android Development Tools(ADT)，这个插件为用户提供了一个强大的综合环境，用于开发 Android 应用程序。ADT 扩展了 Eclipse 的功能，可以让用户快速地建立 Android 项目，创建应用程序界面，在基于 Android 框架 API 的基础上添加组件，以及用 SDK 工具集调试应用程序，甚至导出签名(或未签名)的 APKs 以便发行应用程序。

下面介绍安装 ADT 插件的两种方法。

1. 手动安装 ADT 插件

在 Eclipse 中进行 ADT 插件包手动配置，在下载 Android SDK 压缩文件时已包含相兼容的 ADT 插件，其安装的具体操作步骤如下：

(1) 将“ADT.zip”文件解压，并重新启动 Eclipse，然后选择“窗口”菜单中的“首选项”命令，可见在弹出的“首选项”对话框的左边多了 Android 选项，如图 1-15 所示。

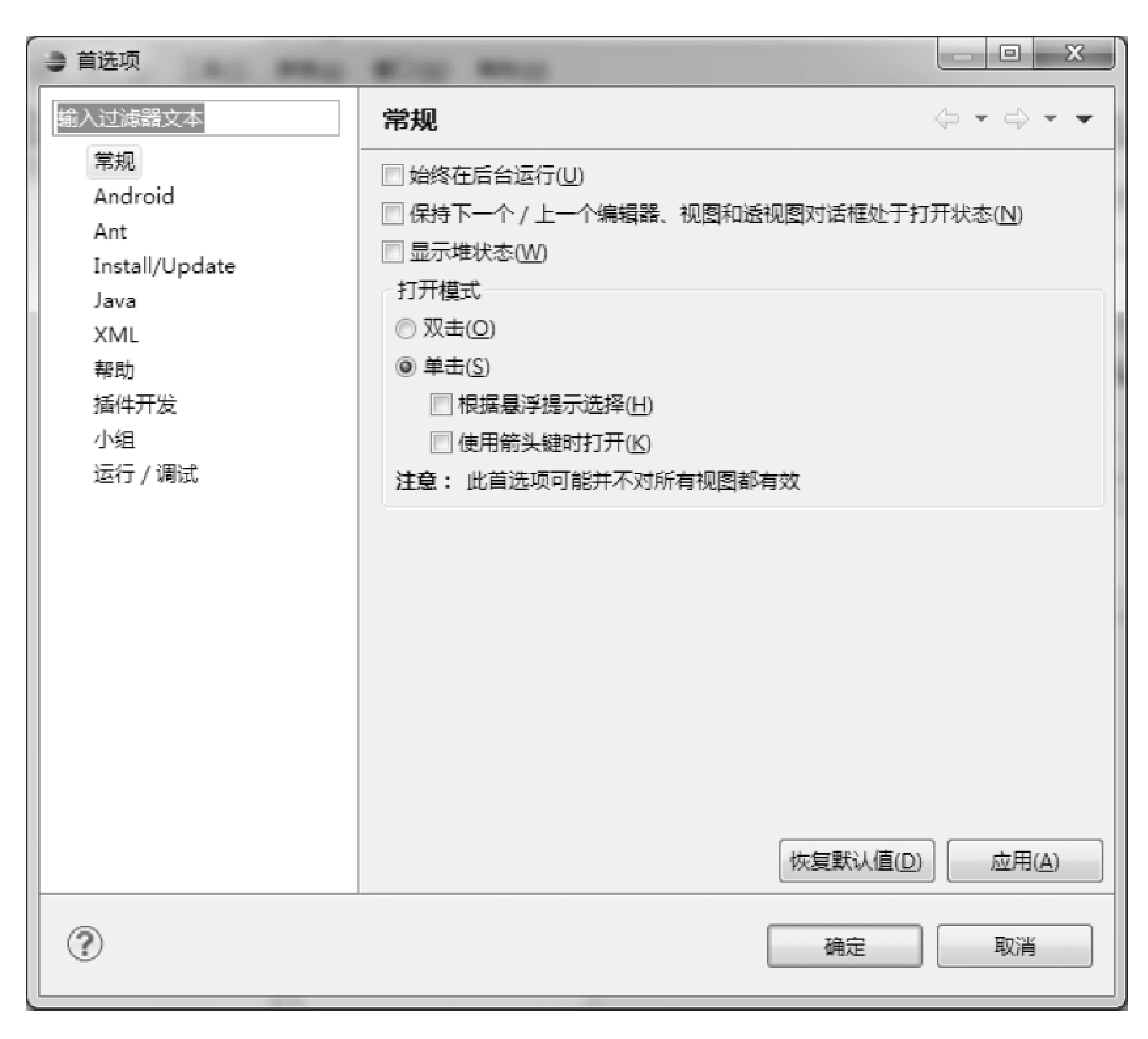


图 1-15 “首选项”对话框

(2) 单击 Android 选项，在该对话框右边的 SDK Location 文本框中设置 Android SDK 的安装路径，这里设置为“D:\Andoridtool\Android_SDK_windows\sdk\sdk”，此时对话框中会列出当前可用的 SDK 版本和 Google API 版本，如图 1-16 所示。至此，即完成了 Eclipse 开发环境下 ADT 插件的安装。

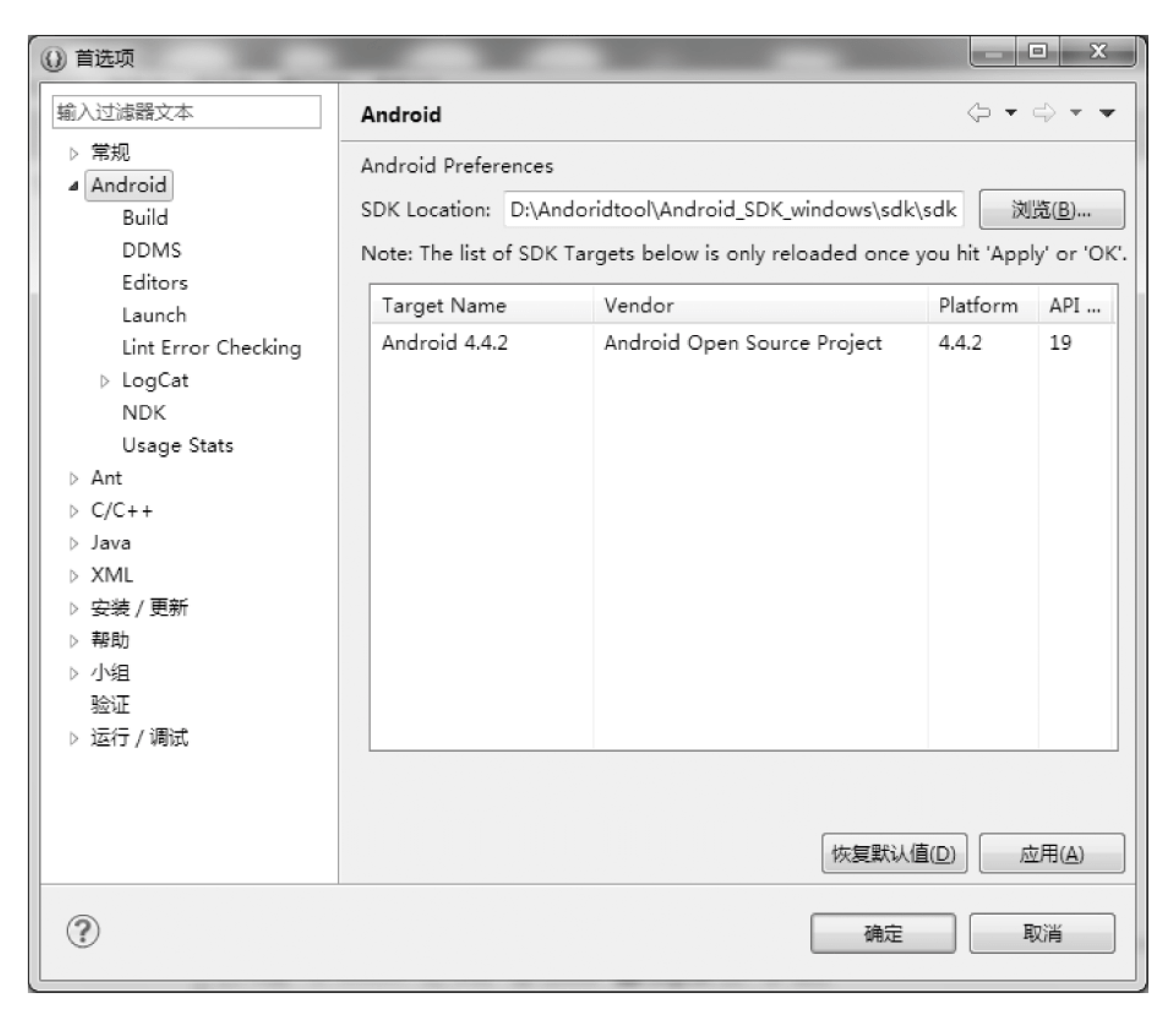


图 1-16　设置 Android SDK 的安装路径

2. Eclipse 在线安装 ADT

除了手动设置 ADT 插件外，还可以采用更简单的在线更新 ADT 插件的方法，具体操作步骤如下：

(1) 打开 Eclipse，然后选择“帮助”菜单中的“安装新软件”选项，如图 1-17 所示。

(2) 在弹出的对话框中单击“添加”按钮，如图 1-18 所示。

(3) 在弹出的 Add Repository 对话框中输入名字和地址，名字可自己命名，例如“abc”，但是在 Location 文本框中必须输入插件的网络地址“http://dl-ssl.google.com/Android/eclipse”(如图 1-19 所示)，然后单击“确定”按钮。

(4) 单击图 1-19 中的“确定”按钮后，会弹出“安装”对话框显示系统中可用的插件，如图 1-20 所示。



图 1-17　添加插件(1)

(5) 选中图 1-20 中的 NDK Plugins 和 Developer Tools 复选框，然后单击“下一步”按钮，进入如图 1-21 所示的界面。

(6) 单击图 1-21 中的“完成”按钮，即开始进行安装。



图 1-18　添加插件(2)

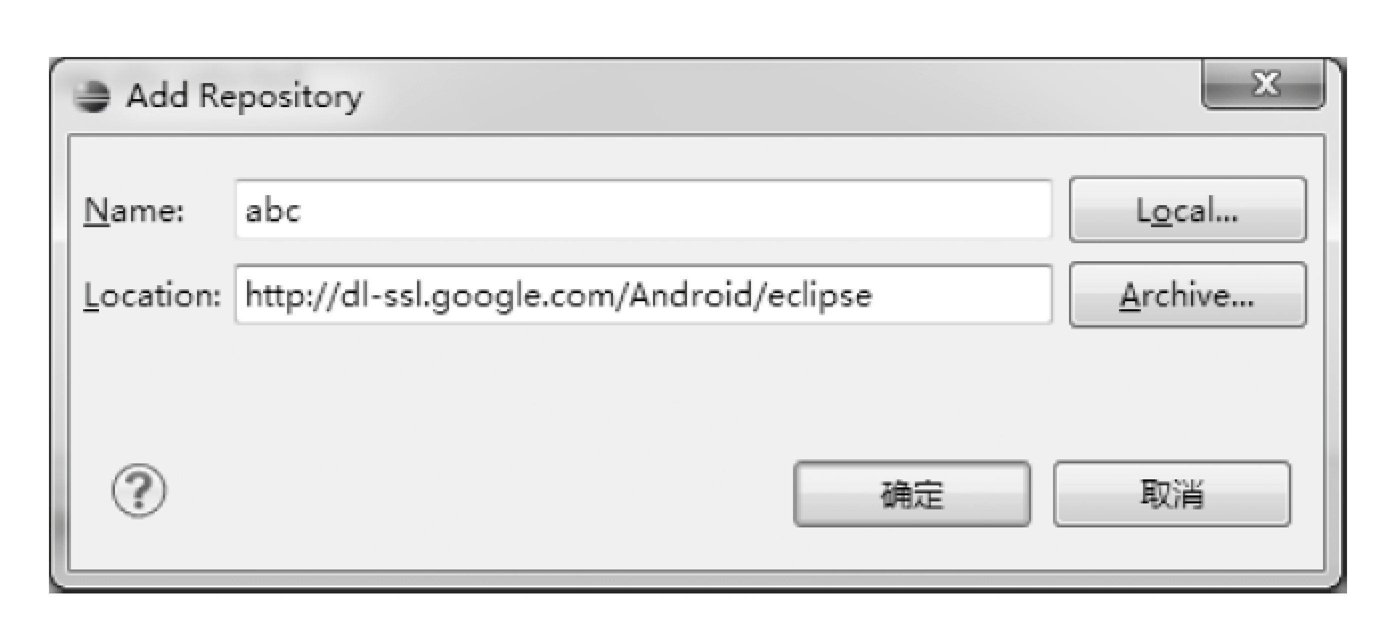


图 1-19　设置插件的网络地址

注意：在上一步骤中，可能会发生计算插件占用资源的情况，过程有点慢。完成后会提示重启 Eclipse 来加载插件，在重启后就可以用了。不同版本的 Eclipse 安装插件的方法和步骤略有不同。



图 1-20　插件列表

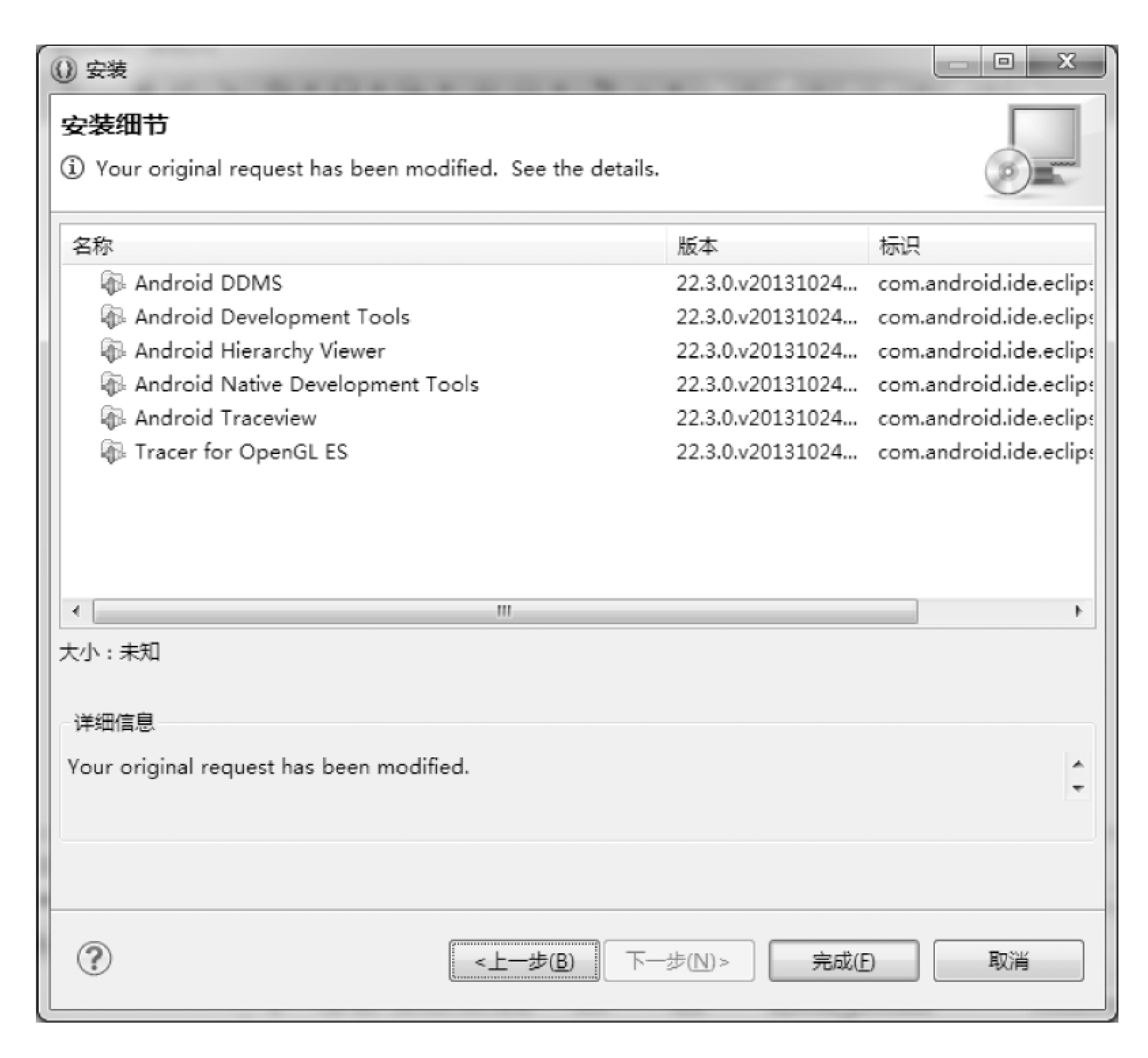


图 1-21　插件安装界面

1.2.6 Android 虚拟设备

AVD 的全称为 Android Virtual Device，它是 Android 运行的虚拟设备，是 Android 的模拟器识别。建立的 Android 要运行，必须创建 AVD，每个 AVD 上可以配置很多运行项目。在创建 AVD 时，可以配置的选项有模拟影像大小、触摸屏、轨迹球、摄像头、屏幕分辨率、键盘、GSM、GPS、Audio 录放、SD 卡支持、缓存大小等。创建 AVD 的方法有两种，一是通过 Eclipse 开发环境创建，二是通过命令行创建。

1. 通过 Eclipse 开发环境创建

其实现步骤如下：

(1) 在图 1-12 中选择 Tools→Manager AVD，即可启动 Android AVD，打开如图 1-22 所示的 Android Virtual Device Manager 窗口。

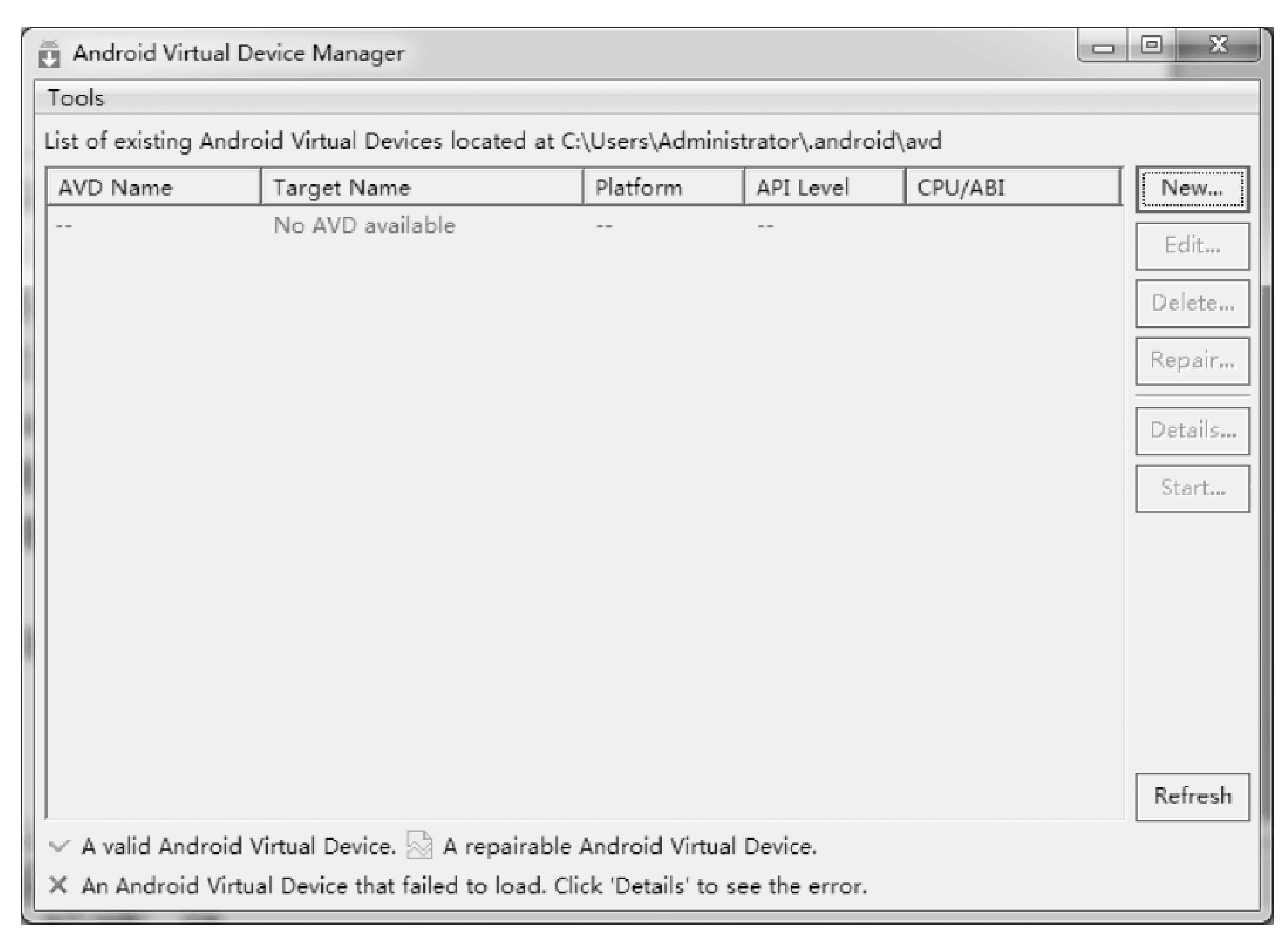


图 1-22　Android Virtual Device Manager 窗口

(2) 单击图 1-22 右侧的 New 按钮，弹出 Create new Android Virtual Device (AVD)对话框，如图 1-23 所示。在该对话框中可以设置模拟器的配置，包括以下几项。

- AVD Name：创建 AVD 的名称。可以在文本框中输入所要创建的 AVD 的名称，注意名称中不能有空格符。
- Target：选择 Android 版本和 API 的等级。单击右边的下拉按钮，选择相应的 Android 版本和 API 的等级。
- SD Card：设置 SD 卡。在 Size 文本中指定 SD 卡的大小。另外，也可以在 File 文本框中设置已有的 SD 卡镜像文件的路径。
- Skin：设置模拟器的外观和屏幕分辨率。单击“Built-in”右边的下拉按钮，可以选择默认的 HVGA(320×480)、QVGA(240×320)、WVGA(480×800 或 480×854)、WQVGA(240×400 或 240×320)几种，在此选择默认的 HVGA(320×480)。另

外，单击“Resolution”项，还可以自定义分辨率。不同版本的 Android 所设置的 Skin 参数有所不同。

- Hardware：设置模拟器支持的硬件设备的属性，包括影像大小、触摸屏、轨迹球、摄像头、屏幕分辨率、键盘、GSM、GPS、Audio 录放、SD 卡支持、缓存区大小等。单击该区域右边的 New 按钮，在弹出的对话框中可以设置各项的属性。

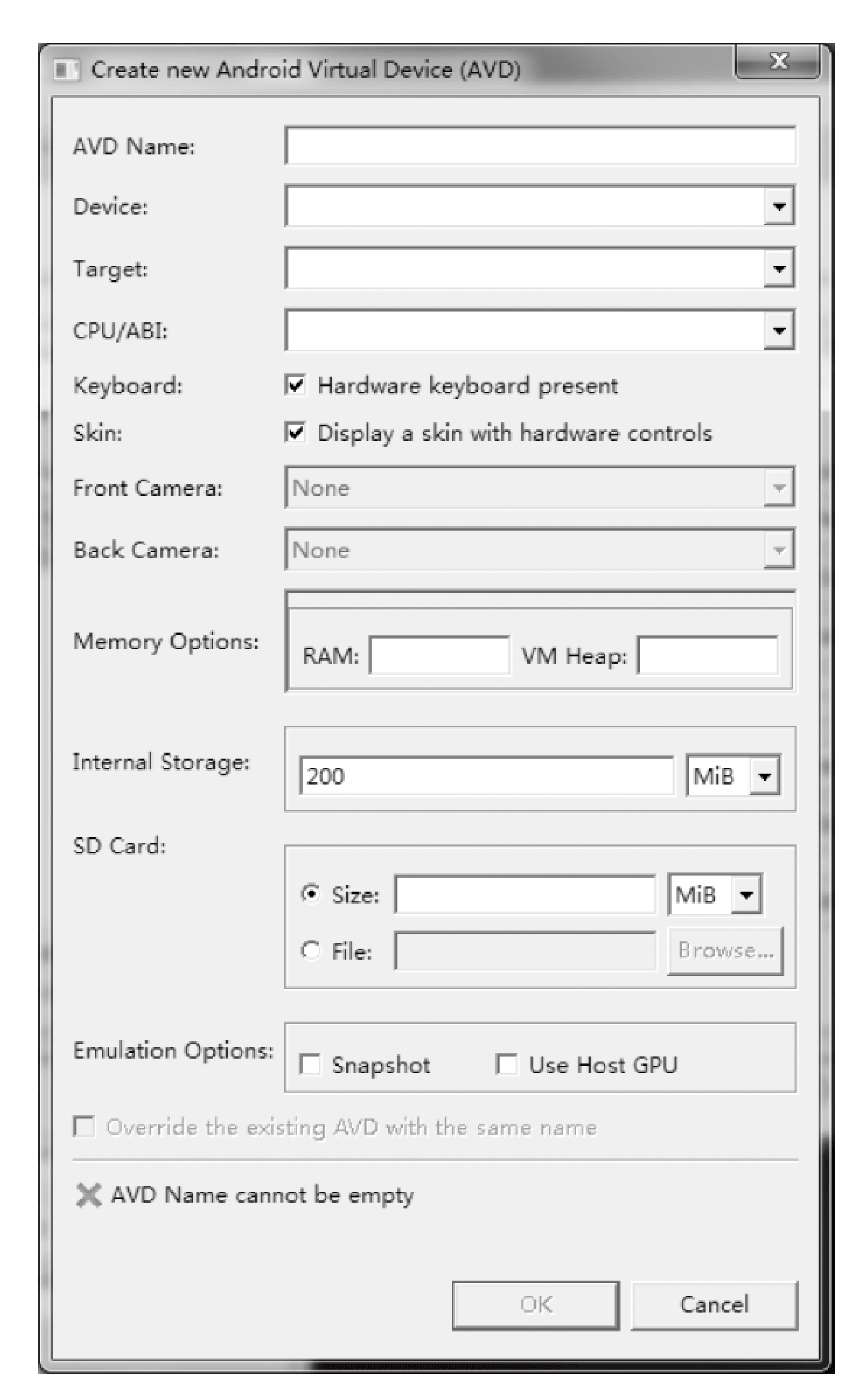


图 1-23　新建 AVD 时的 Emulate 设置

(3) 设置好模拟器的参数后，单击图 1-23 下边的 OK 按钮即可创建一个 AVD。创建好的 AVD 将会显示在如图 1-24 所示的 Android Virtual Device Manager 窗口的文件列表中。

至此，已经创建了一个 Android 模拟器，使用同样的操作可以根据需要创建多个 AVD 模拟器。这样做的好处是，可以模拟程序在不同的 Android 版本上运行的兼容性。

2. 通过命令行创建

通过命令行创建 AVD 的实现步骤如下：

(1) 在 CMD 下输入“android list targets”，查看可用的 Android 平台，如图 1-25 所示。

(2) 按照以下格式创建 AVD：

```
android create avd -- target 2 -- name my_avd
```

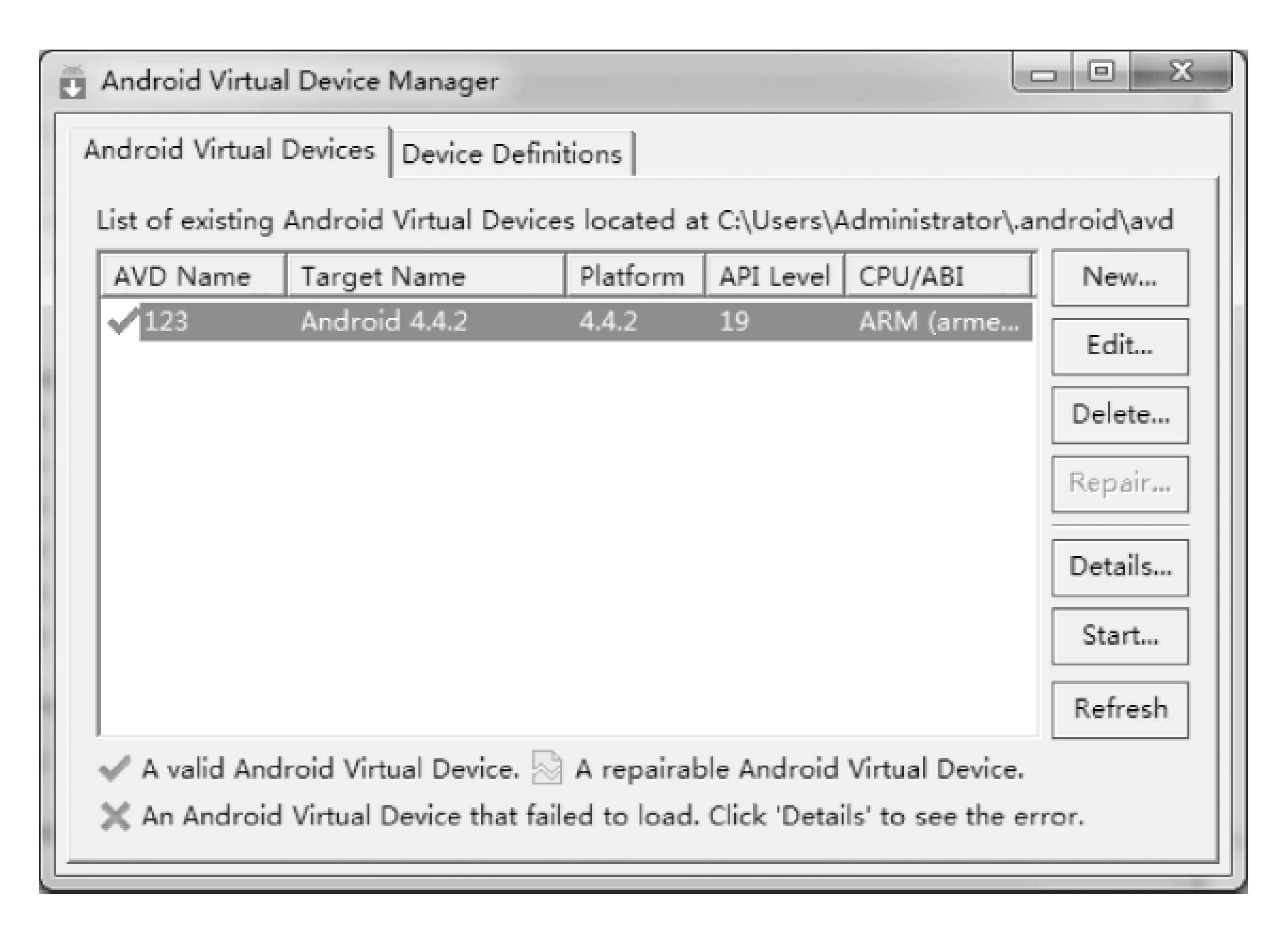


图 1-24　创建新的 AVD

```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7600]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>android list targets
Available Android targets:
----------
id: 1 or "android-19"
     Name: Android 4.4.2
     Type: Platform
     API level: 19
     Revision: 2
     Skins: WVGA800 (default), WVGA854, WXGA800-7in, WXGA720, HVGA, WQVGA432, W
VGA, WQVGA400, QVGA, WXGA800
     ABIs : armeabi-v7a

C:\Users\Administrator>
```

图 1-25　CMD 界面

其中，android 是命令，后面是参数，create avd 指创建 AVD，target 2 是等级，name 是 AVD 的名称。

这里创建名为 my_avd 的 Android 模拟器，如图 1-26 所示。

```
管理员: C:\Windows\system32\cmd.exe

       Usage:
       android [global options] create avd [action options]
       Global options:
  -h --help       : Help on a specific command.
  -v --verbose    : Verbose mode, shows errors, warnings and all messages.
     --clear-cache: Clear the SDK Manager repository manifest cache.
  -s --silent     : Silent mode, shows errors only.

                     Action "create avd":
  Creates a new Android Virtual Device.
Options:
  -c --sdcard   : Path to a shared SD card image, or size of a new sdcard for
                  the new AVD.
  -n --name     : Name of the new AVD. [required]
  -a --snapshot: Place a snapshots file in the AVD, to enable persistence.
  -p --path     : Directory where the new AVD will be created.
  -f --force    : Forces creation (overwrites an existing AVD)
  -s --skin     : Skin for the new AVD.
  -t --target   : Target ID of the new AVD. [required]
  -b --abi      : The ABI to use for the AVD. The default is to auto-select the
```

图 1-26　通过命令行创建 AVD

模拟器可以运行大部分的应用程序,但是实际操作中大部分时间是在真正机器上高速运行,那样效果和效率更高。

1.2.7 运行 AVD

创建好 AVD 后,运行 Android 模拟器有两种方式,其中一种是在 Android Virtual Device Manager 窗口中选中已创建的 AVD,然后单击右侧的 Start 按钮,弹出如图 1-27 所示的 Launch Options 对话框。

单击 Launch Options 对话框中的 Launch 按钮即可成功启动 AVD,如图 1-28 所示。

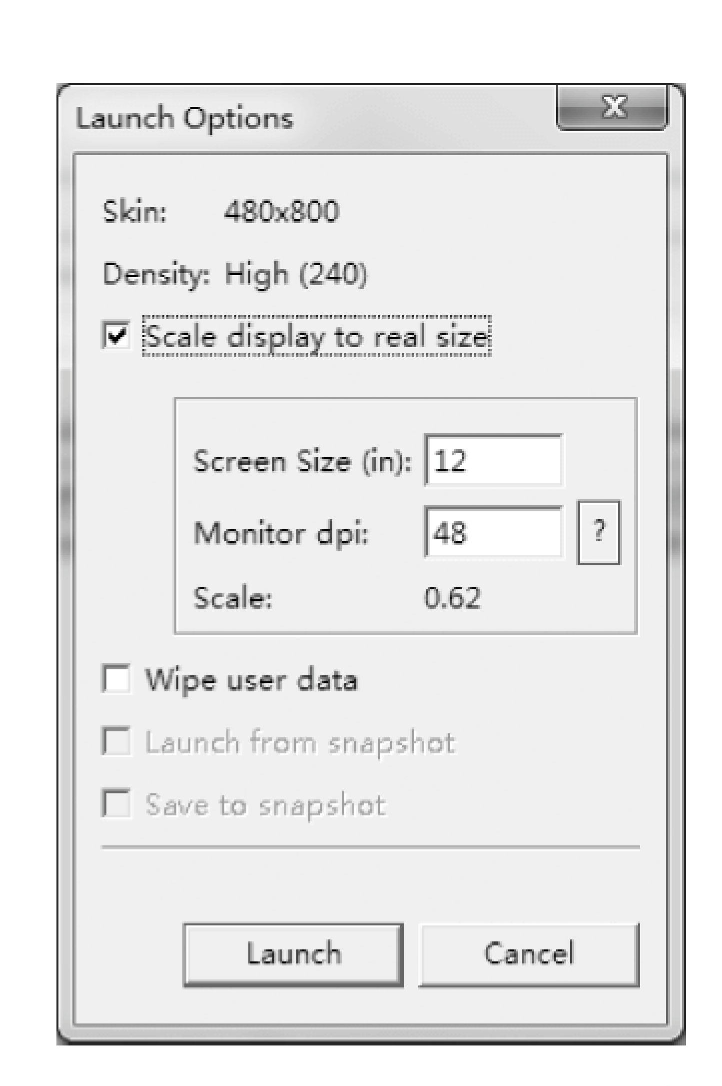


图 1-27 Launch Options 对话框



图 1-28 启动 AVD

在图 1-28 中,右上角的各个控制按钮的名称及功能如表 1-2 所列。

表 1-2 AVD 的控制按钮及功能

模拟器 AVD 的模拟按钮	相应的图标	功 能
音量减小按钮		控制音量大小
电源按钮		设置电话模式,AVD 开关
音量增加按钮		控制音量大小
上/下/左/右按钮		确定按钮
中心按钮		上/下/左/右移动焦点
Home 按钮		返回主界面
Menu 按钮		打开应用程序菜单
查询按钮		在手机内部或上网查询
返回按钮		返回上一级界面

1.3 Android 应用项目组成

Android 的应用项目主要由以下部分组成。

(1) src 文件夹：项目源文件都保存在这个文件夹中。

(2) R.java 文件：这个文件是 Eclipse 自动生成的，应用开发者不需要去修改里面的内容。

(3) Android Private Libraries：这是应用运行的 Android 库。

(4) assets 文件夹：里面主要放置多媒体等一些文件。

(5) res 文件夹：主要放置应用会用到的资源文件。

(6) drawable：主要放置应用会用到的图片资源。

(7) layout：主要放置会用到的布局文件。这些布局文件都是 XML 文件。

(8) values：主要放置字符串(strings.xml)、颜色(colors.xml)、数组(arrays.xml)。

(9) AndroidManifest.xml：相当于应用的配置文件。在这个文件中，必须声明应用的名称，应用所用到的 Activity、Service 以及 Receiver 等。

在 Eclipse 中，一个基本的 Android 项目的目录结构如图 1-29 所示。

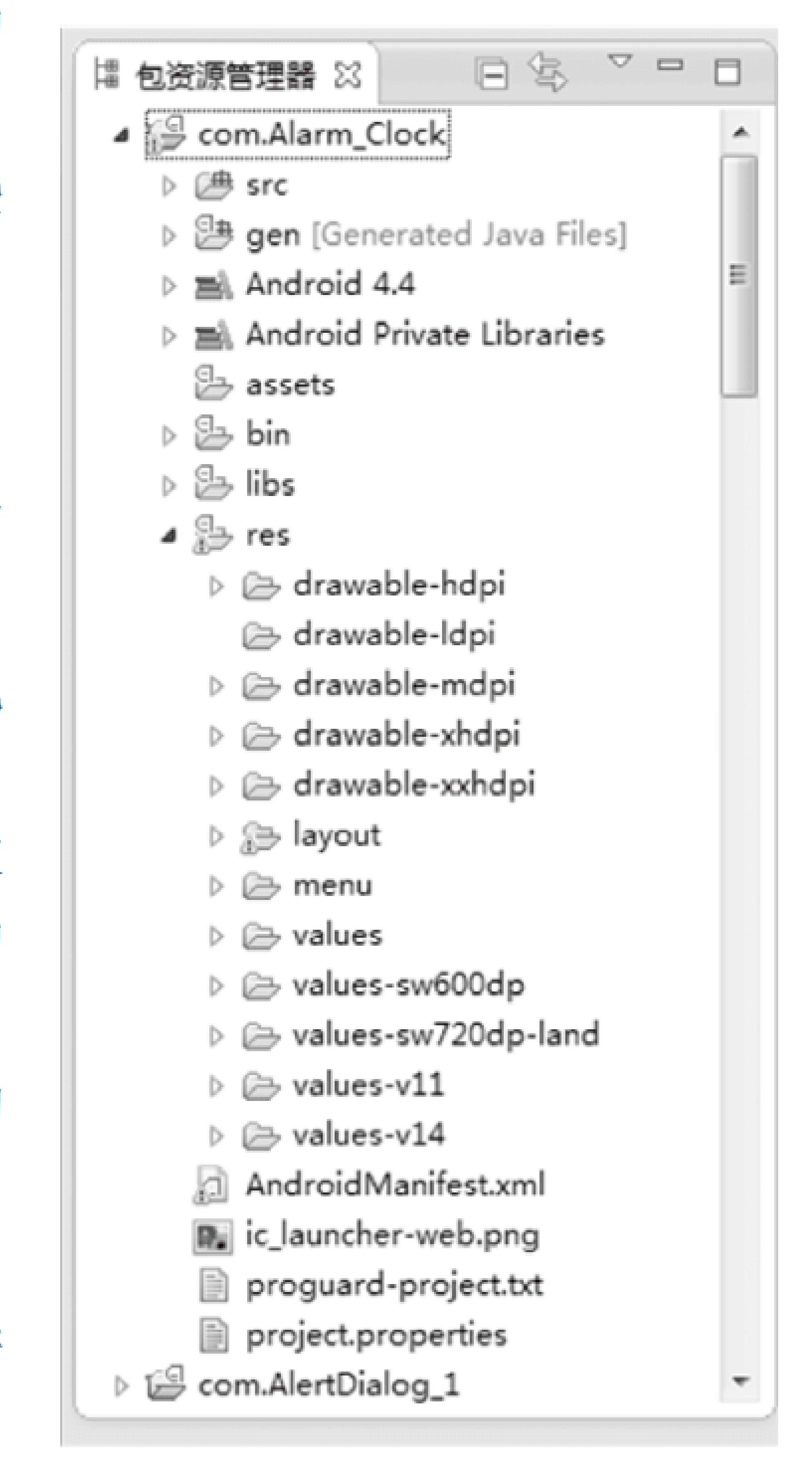


图 1-29 Android 应用工程文件组成

1. src

与一般的 Java 项目一样，src 下保存的是项目的所有包及源文件(.java)，res 下包含了项目中的所有资源，例如，程序图标(drawable)、布局文件(layout)和常量(value)等。不同的是，在 Java 项目中没有 gen 文件夹，也没有每个 Android 项目都必须有的 AndroidManifest.xml 文件。

“.java”格式文件是在建立项目时自动生成的，这个文件是只读模式，R.java 文件是定义该项目所有资源的索引文件。Helloworld 项目的 R.java 文件的代码如下：

```
package fs.helloworld;
public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int activity_horizontal_margin = 0x7f040000;
        public static final int activity_vertical_margin = 0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher = 0x7f020000;
    }
    public static final class id {
```

```
        public static final int action_settings = 0x7f080000;
    }
    public static final class layout {
        public static final int main = 0x7f030000;
    }
    public static final class menu {
        public static final int main = 0x7f070000;
    }
    public static final class string {
        public static final int action_settings = 0x7f050001;
        public static final int app_name = 0x7f050000;
        public static final int hello_world = 0x7f050002;
    }
    public static final class style {
        public static final int AppTheme = 0x7f060001;
    }
}
```

从上述代码中可以看到，文件中定义了很多常量，并且会发现这些常量的名字都与 res 文件夹中的文件名相同，这再次证明 R. java 文件中存储的是该项目所有资源的索引。有了这个文件，在程序中使用资源将变得更加方便，可以很快地找到要使用的资源。由于这个文件不能手动编辑，所以当用户在项目中加入了新的资源时，只需要刷新一下该项目，R. java 文件便自动生成了所有资源的索引。

2. res

在 res 下包含了该项目所用到的资源文件，这里面的每一个文件或者资源都将在 R. java 文件中进行索引定义。主要包括以下几类。

- 图片文件：分别提供了高分辨率(drawable-hdpi)、低分辨率(drawable-ldpi)、中分辨率(drawable-mdpi)、超高分辨率(drawable-xhdpi)、超高清分辨率(drawable-xxhdpi)的图片文件。
- 布局文件：在 layout 文件夹下，默认只有一个 main. xml，用户也可以添加更多的布局文件。
- 字符串：在 values 文件夹下的 strings. xml 文件中。

打开 main. xml 布局文件，代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

在该布局文件中首先定义了相对布局，内部只有一个文本框控件。这个控件显示了内容引用了 string.xml 文件中的 hello 变量。

其中，

- <RelativeLayout></RelativeLayout>：相对版面配置，在这个标签中，所有元件都是按相对排队排成的。
- android:layout_width：定义当前视图在屏幕上所占的宽度，fill_parent 表示填充整个屏幕。
- android:layout_height：随着文字栏位的不同而改变这个视图的宽度或高度。
- android:paddingBottom：指屏幕界面底部的填充方式。
- android:paddingLeft：指屏幕界面左侧的填充方式。
- android:paddingRight：指屏幕界面右侧的填充方式。
- android:paddingTop：指屏幕界面顶部的填充方式。
- tools:context：该布局文件所调用的 Activity 内容。

在上述布局代码中，使用了一个 TextView 来配置文件标签 Widget（构件），其中设置的属性 android:layout_width 为整个屏幕的宽度，android:layout_height 可以根据文字来改变高度，而 android:text 则设置了这个 TextView 要显示的文字内容，这里引用了 @string 中的 hello 字符串，即 string.xml 文件中的 hello 所代表的字符串资源。hello 字符串的内容“HelloWorld、HelloAndroid”就是用户在运行 HelloAndroid 项目时看到的字符串。

strings.xml 文件的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name"> Hello World </string>
    <string name = "action_settings"> Settings </string>
    <string name = "hello_world"> Hello world! </string>
</resources>
```

3. AndroidManifest.xml 文件

在文件 AndroidManifest.xml 中包含了该项目中所使用的 Activity、Service、Receiver，以下代码为 HelloWorld 项目中的 AndroidManifest.xml 文件。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"    //根结点
    package = "fs.helloworld"                          //包名
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />            //SDK 版本
    <application                                      //图标和应用程序名称
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <activity
            android:name = "fs.helloworld.MainActivity" //默认启动的 Activity
```

```
            android:label = "@string/app_name" >         //Activity 名称
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </activity>
    </application>
</manifest>
```

1.4 第一个 Android 实例

在 Eclipse 中，可以非常便捷地创建、调试 Android 应用程序。下面创建一个最基本的 Android 项目。

1. 新建工作空间

通常，创建的应用项目都需要存盘，或者使用 C 盘，或者将创建的文件放置到其他盘上，那么怎样做呢？其实现步骤如下：

（1）在 Eclipse 主界面中选择“文件”→“切换工作空间”→“其他”命令，如图 1-30 所示。



图 1-30 切换工作空间操作

（2）弹出“工作空间启动程序”对话框，在“工作空间”右侧的文本框中输入所需要的路径，例如把文件放在 E 盘新建的 ltx_Android1 文件夹中，如图 1-31 所示。

（3）设置好存盘路径后，单击图 1-31 中的“确定”按钮即完成了工作空间的创建，也就是说以后所创建的项目文件都在该文件夹中。

（4）当完成工作空间启动程序设置后，Eclipse 会自动重新启动。

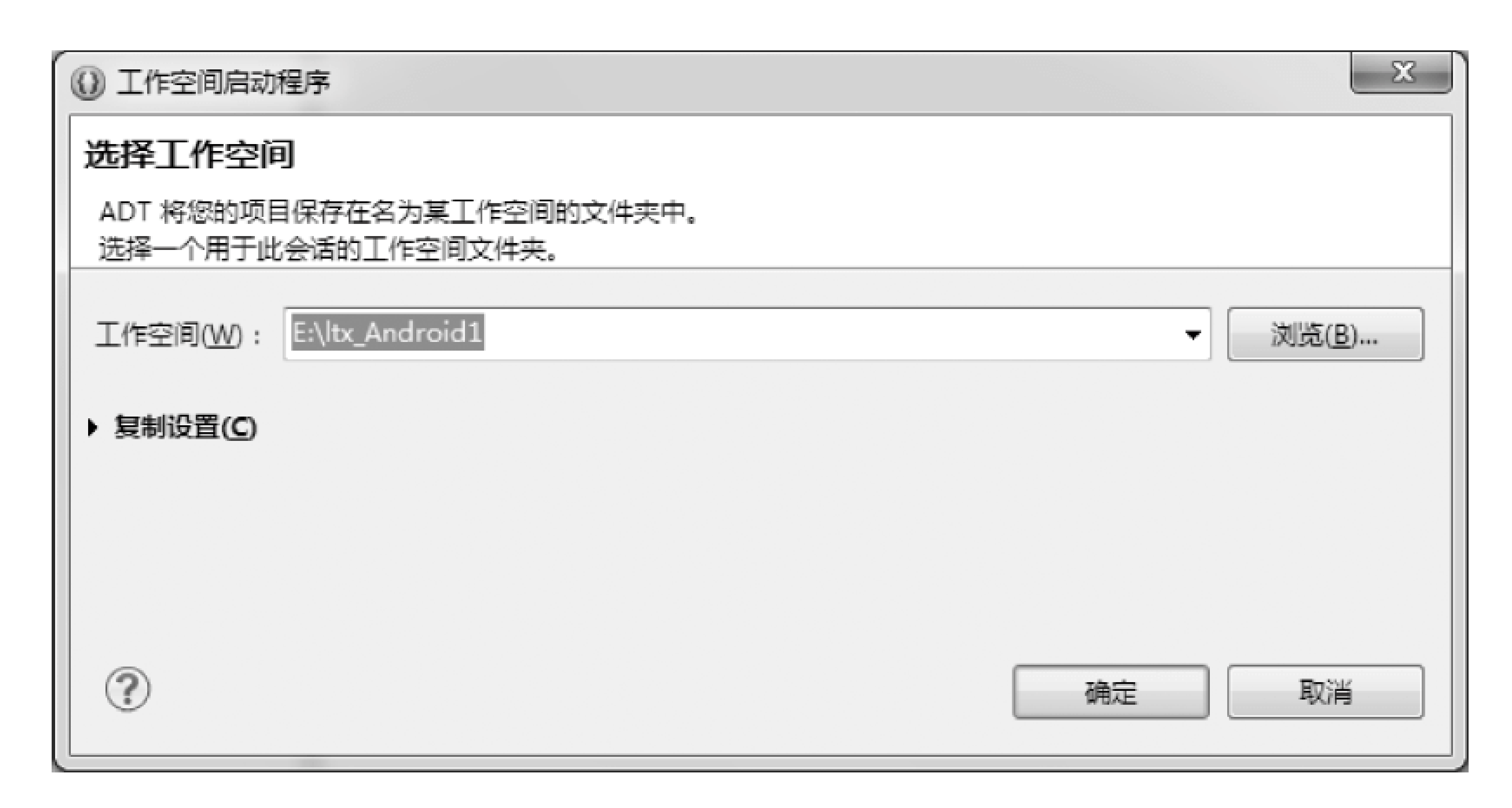


图 1-31 “工作空间启动程序”对话框

2. 创建 Android 项目

(1) 在 Eclipse 主界面中单击“新建”按钮 创建一个 Android 应用项目，Eclipse 会弹出如图 1-32 所示的对话框。

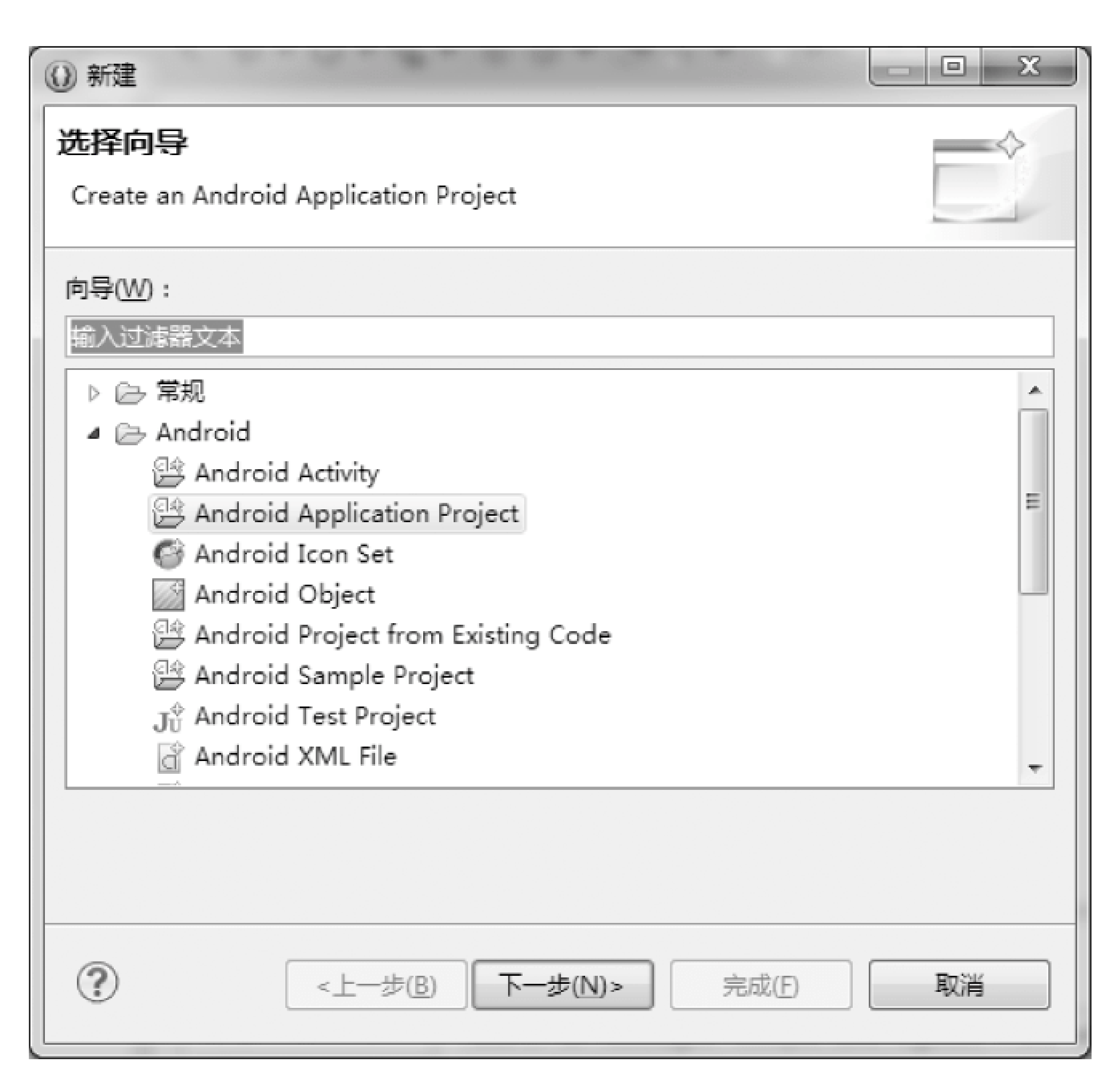


图 1-32 “新建”对话框

(2) 选择图 1-32 中的 Android Application Project 项，然后单击“下一步”按钮，弹出如图 1-33 所示的创建 Android 项目的对话框。

- Application Name：填写应用程序的名称。默认情况下，会将前面填写的项目名称填写在此处，可以进行修改。该名称将作为应用程序的名称出现在手机应用列表中。
- Project Name：填写工程项目的名称，即在 Eclipse 工作空间中创建的文件夹名称，一般以 com. * 形式命名。



图 1-33　创建 Android 项目的对话框

- Package Name：Java 源文件的包名，Eclipse 会自动在 src 下创建该包名。该包名一般以 *.* 形式命名。

(3) 单击图 1-33 中的“下一步”按钮，进入如图 1-34 所示的界面，选择默认值，然后单击“下一步”按钮。

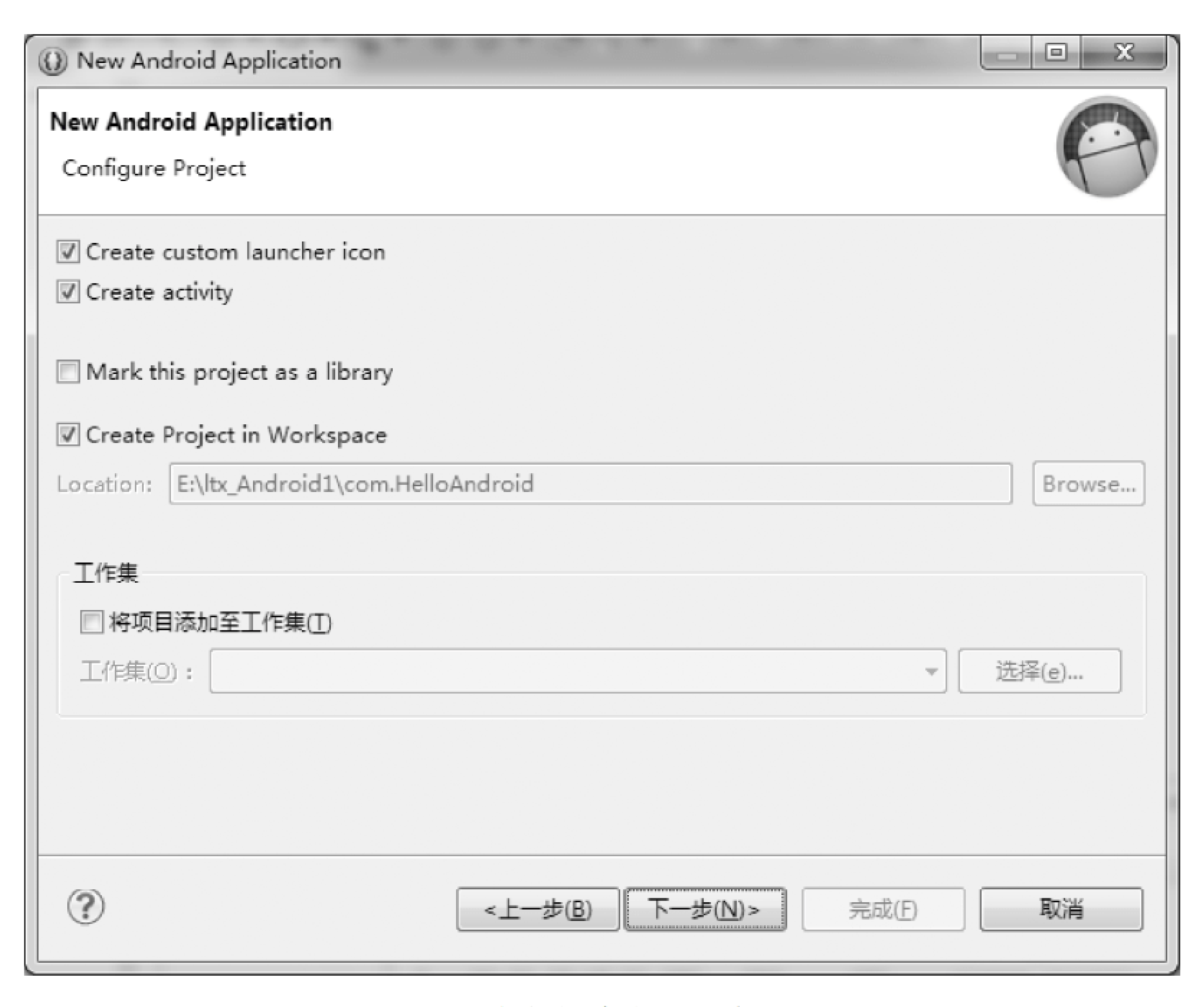


图 1-34　确定创建应用程序的界面

(4) 进入如图 1-35 所示的界面，选择默认值，然后单击“下一步”按钮。

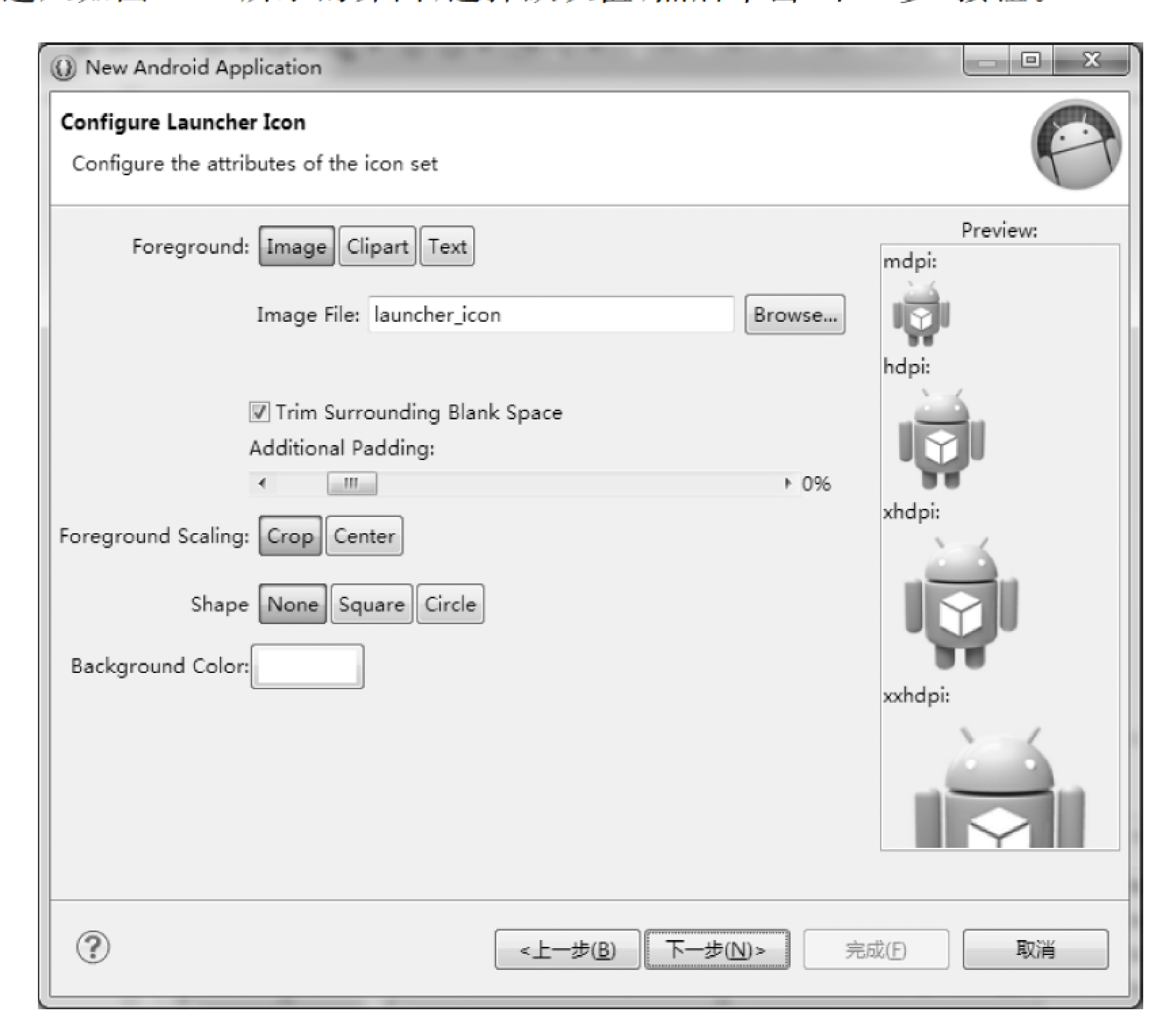


图 1-35　配置发射器图标的界面

(5) 进入如图 1-36 所示的界面，选择默认值，单击“下一步”按钮。



图 1-36　创建活动界面

(6) 进入如图 1-37 所示的界面,将 Layout Name 命名改为 main,其他采用默认值,然后单击“完成”按钮,即可完成 Android 项目的创建。



图 1-37 新的空白活动界面

(7) 在 Eclipse 中成功地创建了一个 Android 项目,Android 项目创建完成后将看到如图 1-38 左侧树形结构所示的项目结构。

(8) Android 项目的 layout 文件夹下有一个 main. xml 文件,该文件用于定义 Android 的应用用户界面。在 Eclipse 工具中打开该文件,将看到如图 1-39 所示的界面。

在图 1-39 所示界面的控件面板中向程序中拖入一个 Button 控件(按钮),然后切换到源代码编写界面,将 main. xml 文件的代码修改为:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

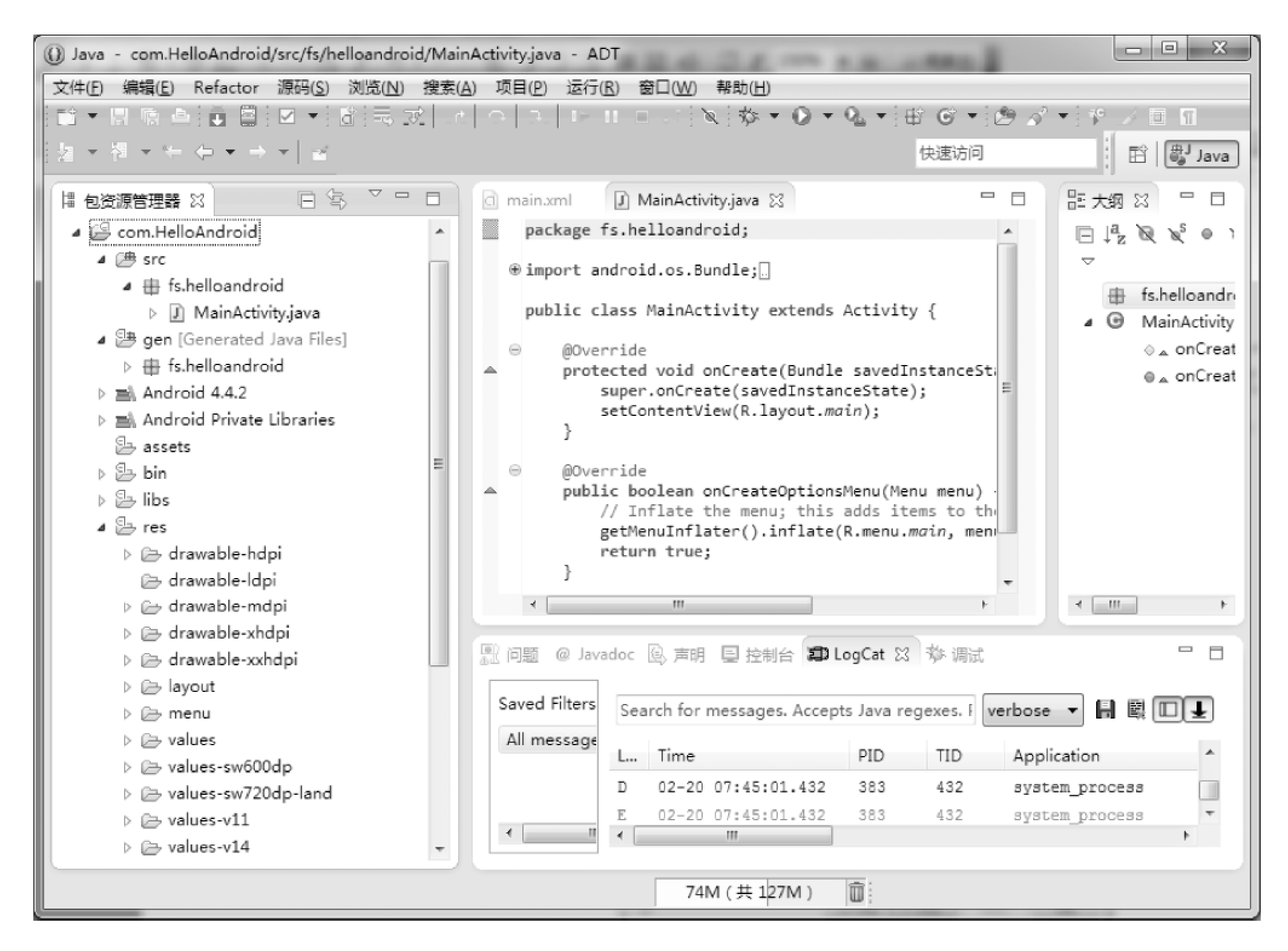


图 1-38　Hello Android 项目结构



图 1-39　ADT 提供的界面设计工具

```
    android:text = "@string/hello_world" />
<Button
    android:id = "@ + id/button1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
```

```
        android:layout_alignParentBottom = "true"
        android:layout_centerHorizontal = "true"
        android:layout_marginBottom = "184dp"
        android:text = "单击" />
</RelativeLayout>
```


Android 之所以把用户界面放在 XML 文档中定义，是为了让 XML 文档专门负责用户 UI 设置，而 Java 程序专门负责业务实现，这样可以降低程序的耦合性。

(9) Android 项目的 src 文件夹用于存放 Android 项目的源代码，该文件夹下有一个 MainActivity. java 文件，它是 Android 项目的 Java 文件。打开该文件，将代码修改为：

```
package fs.helloandroid;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
public class MainActivity extends Activity
{
    //当第一次创建该 Activity 时回调该方法
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        //设置使用 main.xml 文件定义的页面布局
        setContentView(R.layout.main);
        //获取 UI 界面中 ID 为 R.id.ok 的按钮
        Button bn = (Button)findViewById(R.id.button1);
        //为按钮绑定一个单击事件的监听器
        bn.setOnClickListener(new OnClickListener()
        {
            public void onClick(View v)
            {
                //获取 UI 界面中 ID 为 R.id.show 的文本框
                final TextView show = (TextView)findViewById(R.id.textView1);
                //改变文本框的文本内容
                show.setText("Hello Android~" + new java.util.Date());
            }
        });
    }
}
```

以上程序中十分简单，只做了 3 件事：

① 设置该 Activity 使用 main. xml 文件定义的界面布局文件作为用户界面。

② 获取 ID 为 R. id. button1 的按钮。

③ 为上一步获得的按钮绑定事件监听器，即在事件监听器的处理器方法中改变 ID 为 R. id. textView1 的文本框内容。

(10) 为程序添加标题，选择 res\value 下的 string. xml 文件，将代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<resources>
    <string name="app_name">第一个 Android 应用实例</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello Android</string>
</resources>
```

单击 Eclipse 界面中的“运行”按钮 ，或选中程序右击，在弹出的快捷菜单中选择“运行方式→Android Application”命令，即可运行 com. HelloAndroid 项目，效果如图 1-40 所示。



图 1-40 Hello Android 显示界面

3. DDMS 的使用

在 Android SDK 工具中提供了 DDMS(Dalvik Debug Monitor Service)对 Android 的应用程序进行调试和模拟服务，主要提供了针对特定的进程查看正在运行的线程以及堆信息、输入日志(Logcat)、广播状态信息、模拟电话呼叫、接收 SMS、虚拟地理坐标、为测试设备截屏等。

DDMS 会搭建 Eclipse 本地与测试终端(Emulator 或者真实设备)的连接，它们应用各自建立的端口监听调试器的信息，DDMS 可以实时监测到测试终端的连接情况。当有新的测试终端连接后，DDMS 将捕捉到终端的 ID，并通过 ADB 工具建立调试器，从而实现发送指令到测度终端的目的。

1) 开启 DDMS 视图

在 Eclipse 的右上角有一个 DDMS 图标，单击该图标即可打开 Eclipse 中所有的视图界面。除此之外，还可以在 Eclipse 的菜单栏中选择“窗口→打开透视图→其他”命令，打开所有视图，如图 1-41 所示。选择图 1-41 中的 DDMS，切换到 DDMS 界面。

2）DDMS 的功能

在 DDMS 视图界面中有调试 Android 设备经常使用的工具，主要包括设备（Devices）、模拟器控制台（Emulator Control）、日志输出（LogCat）、文件目录（File Explorer）以及线程、堆栈等。如果在 DDMS 界面中没有找到这些功能选项，则在 Eclipse 的菜单栏中选择"窗口→显示视图→其他"命令开启，如图 1-42 所示。

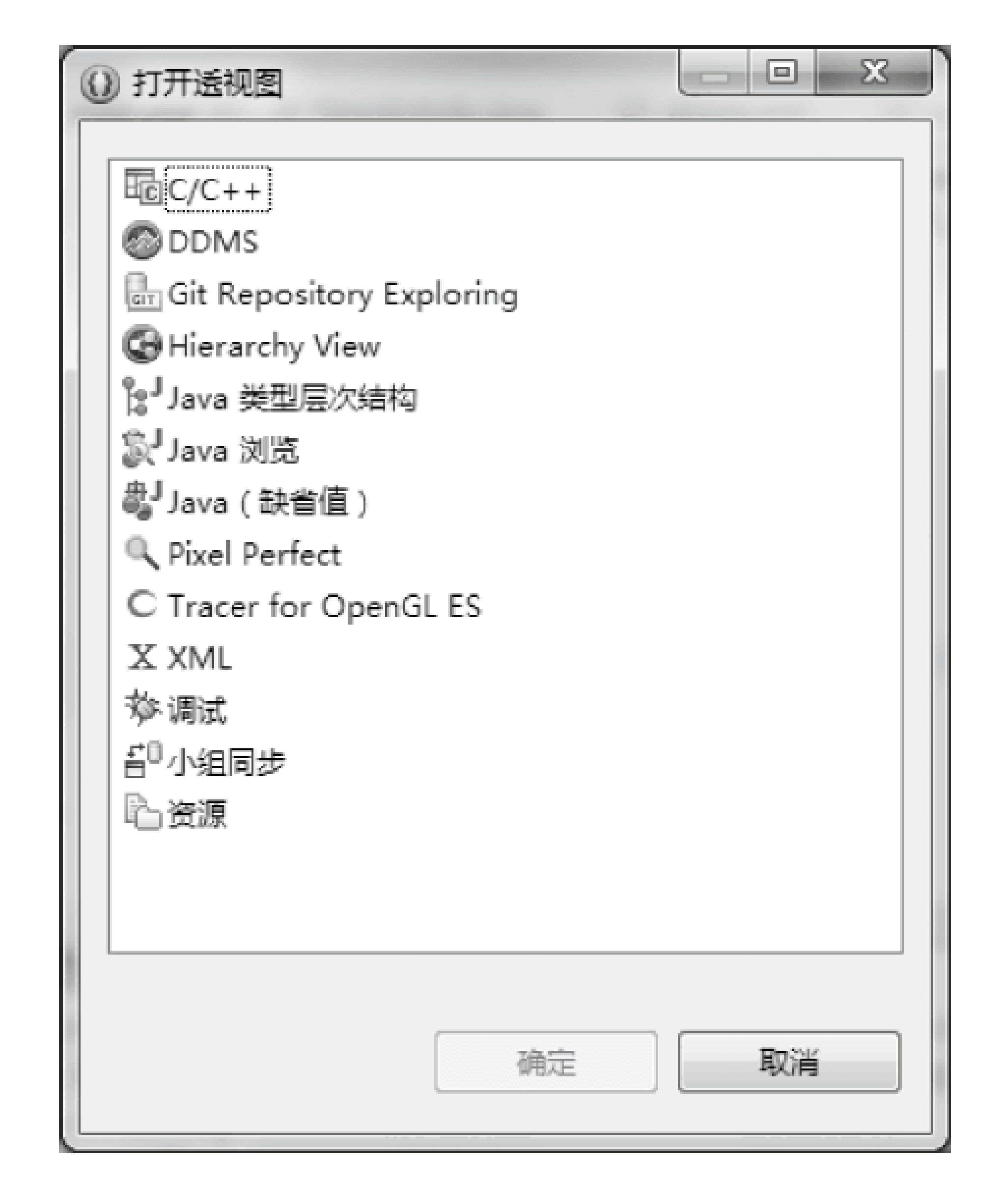


图 1-41　打开透视图

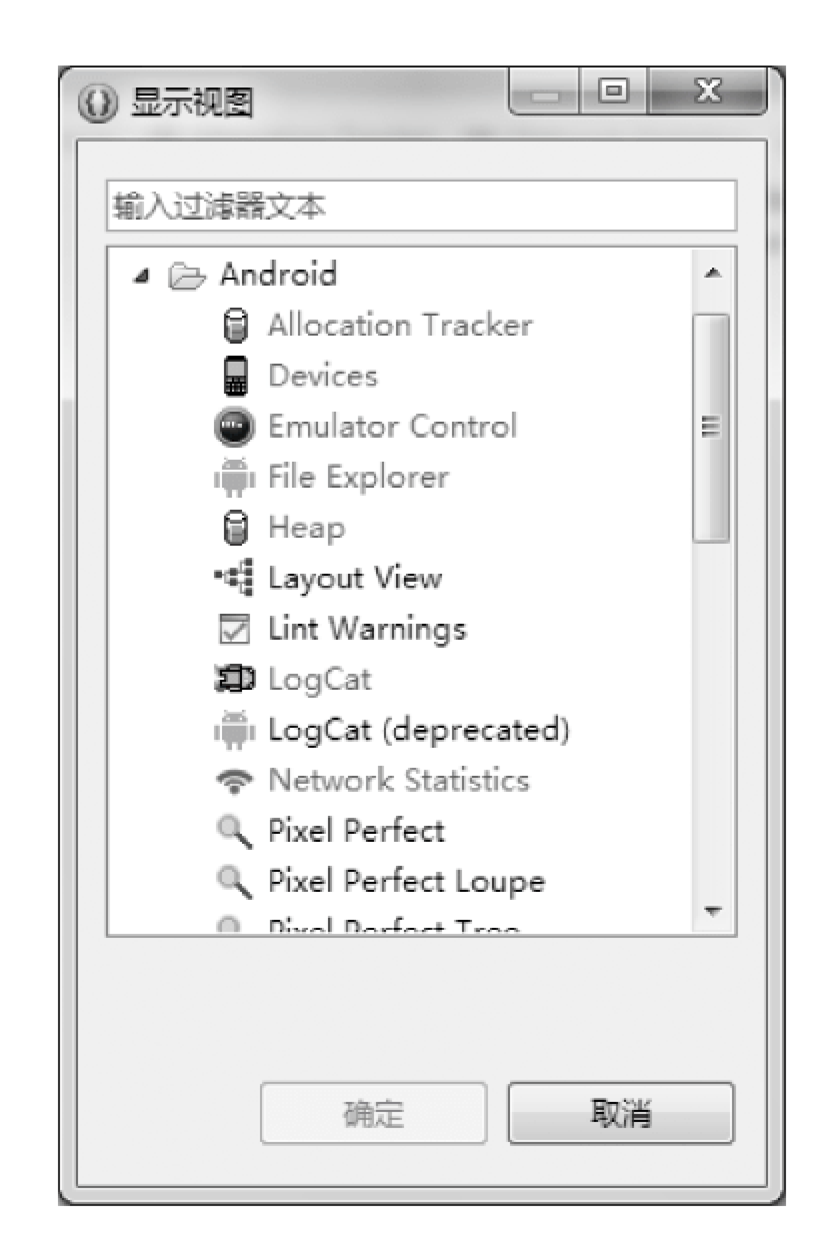


图 1-42　显示视图

在 DDMS 提供的功能中，最常用的功能有以下 4 个。

（1）设备（Devices）：设置功能视图一般在 DDMS 的左上角，其标签为 Devices，如图 1-43 所示。在该视图中显示所有连接的 Android 设备，并且详细列出该 Android 设备中可连接调试的应用程序进程。从该图可以看出列表中从左到右分别为应用程序名以及调试器连接的端口号。在进行调试时，一般只需要关心应用程序名。

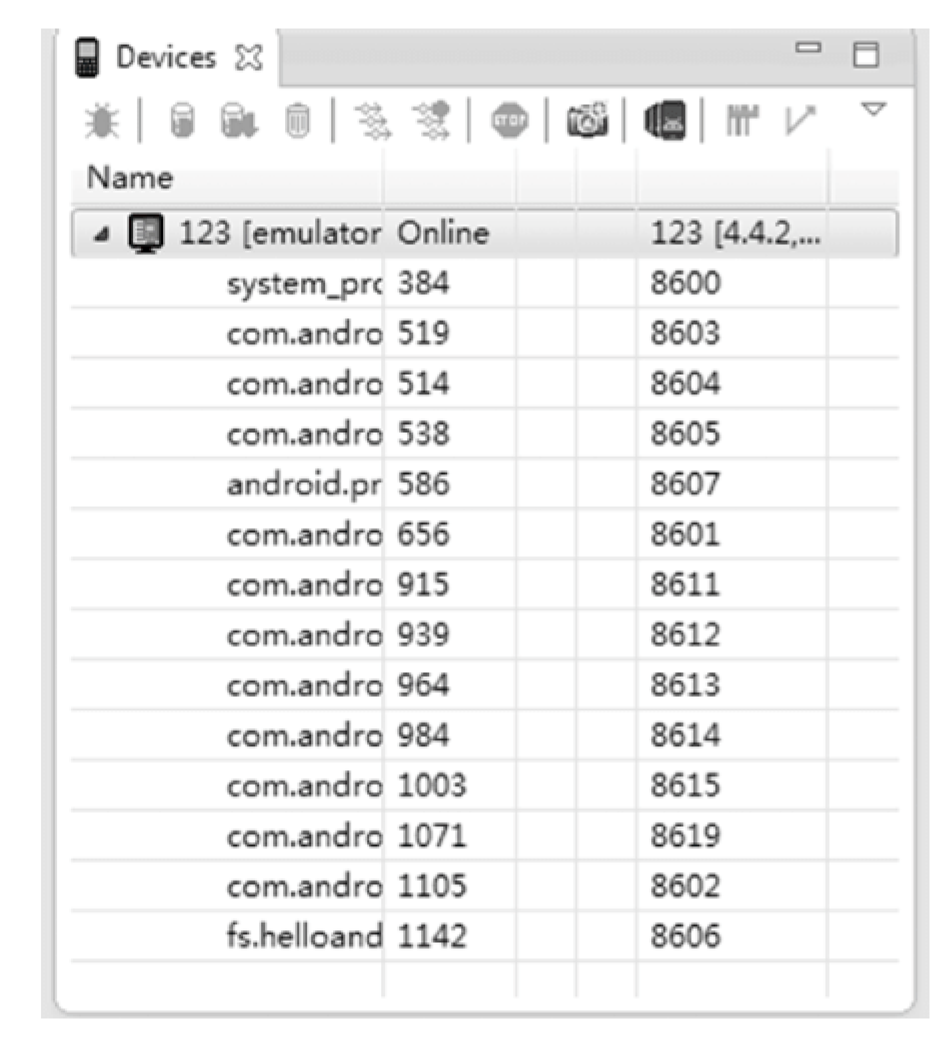


图 1-43　设备列表

当选择了列表中的某一个应用程序时，在视图的右上角有一排功能按钮可使用。它们主要用于调试某个应用，主要的功能有调试选项（Debug the selected process）、线程查看（Update Threads）、堆栈查看（Update Heap）、终止进程（Stop Process）和截屏（ScreenShot）。

- Debug the selected process：用于显示被选择进程与调试器连接状态。如果进程前带有绿色，表示该进程的源文件在 Eclipse 中处于打开状态，并已经开启了调试器监听进程的运行情况。

• Update Threads：用于查看当前进程所包含的线程。选中任意进程，单击该按钮后，被选中的进程名称后边会出现显示线程信息标识，并可以在 Threads 功能界面中看到详细的线程运行情况。
• Update Heap：用于查看当前进程堆栈内存的使用情况。选中任意进程，单击该按钮，可在 Heap 功能界面中看到详细的堆栈使用情况，与 Update Threads 类似。
• Stop Process：终止当前进程。选择进程后，单击该按钮则强制终止了该进程。
• ScreenShot：截取当前测试终端桌面。

(2) 模拟器控制台(Emulator Control)：由于在模拟器中不断直接使用真机的电话、短信、GPS 位置等功能，当使用模拟区测试这些功能时，可以通过该控制台实现对这些交互功能的模拟。模拟器控制台视图如图 1-44 所示。

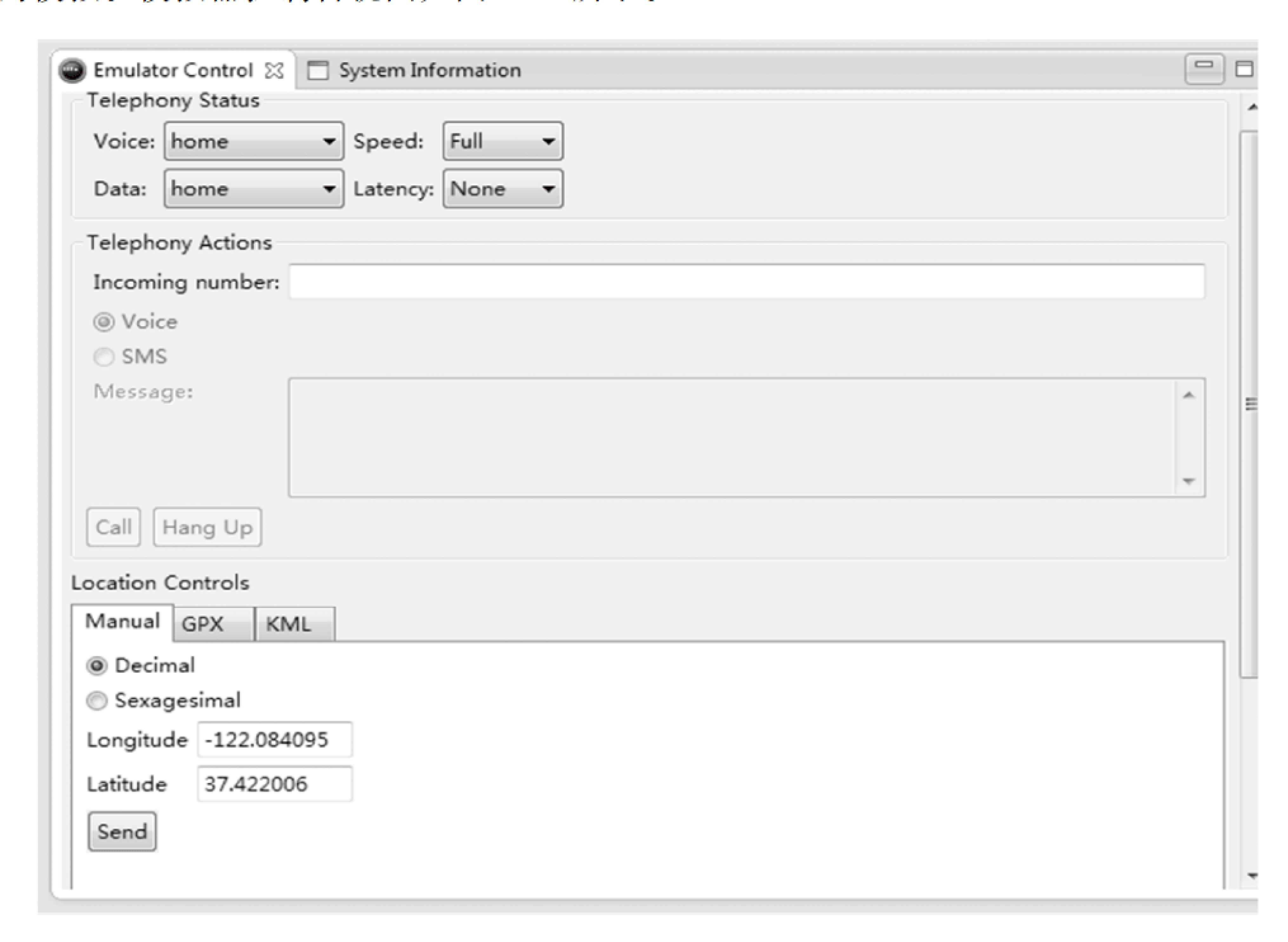


图 1-44　控制台

对其选项说明如下。
• Telephony Status：选择模拟语音质量以及信号连接模式。
• Telephony Actions：模拟电话呼入和发送短信到测试的模拟器。其中，Incoming number 为设置本地呼叫模拟器的号码，Voice 选项表示模拟电话呼入模拟器；SMS 选项表示模拟短信发送到模拟器中。
• Location Controls：模拟地理坐标或模拟动态的路线坐标变化并显示预设的地理标识。其中有 3 个选项卡，表示可以使用不同的 3 种方式，Manually 方式为手动为终端发送二维经纬坐标；GPX 方式为通过 GPX 文件导入序列动态变化地理坐标，从而模拟行 GPS 变化的数值；KML 方式为通过 KML 文件导入独特的地理标识，并以动态形式根据变化的地理坐标显示在测试终端。

(3) 文件目录(File Explorer):在 DDMS 界面的右边占用较大一块区域的便是模拟器运行的详细信息,有多个选项卡,其中,"File Explorer"为文件目录,如图 1-45 所示。

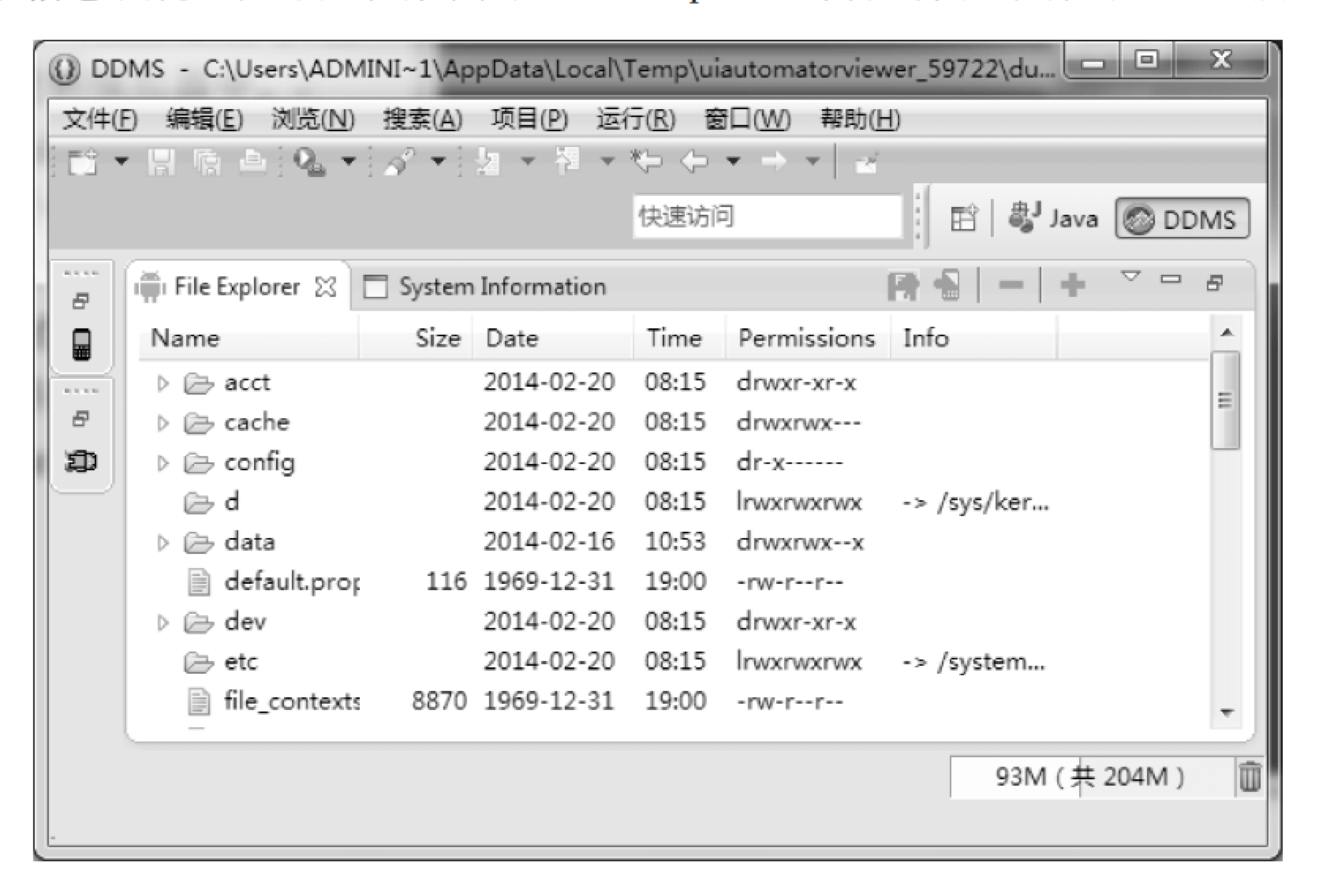


图 1-45 文件目录

在文件目录中显示 Android 设备的文件系统信息。一般情况下,File Explorer 中会有 3 个文件夹,即 data、mnt 和 system。

- data:对应手机的 RAM,存放 Android 系统运行时的 Cache 等临时数据。如果没有 roots 权限,APK 程序将会安装在\data\app 中(只是存放 APK 文件本身),在\data\data 中存放着所有程序(系统应用程序和第三方应用程序)的详细数据目录信息。
- mnt:最重要的是其下的"sdcard",对应于 SD Card 的目录文件。
- system:对应手机的 ROM,存放 Android 系统以及系统自带的应用程序等。

除了可以查看这 3 个文件夹外,还可以使用 File Explorer 对文件进行操作。选项卡右上角的操作按钮从左到右分别为将 Android 设备保存到本地、上传到 Android 设备、删除文件、添加文件夹。当然,在使用这 4 个功能时,需要对 Android 设备的文件系统具有相应的操作权限。

(4) 日志输出(LogCat):在模拟器中的所有输出信息都显示在日志信息中,该视图一般在最下方,如图 1-46 所示。

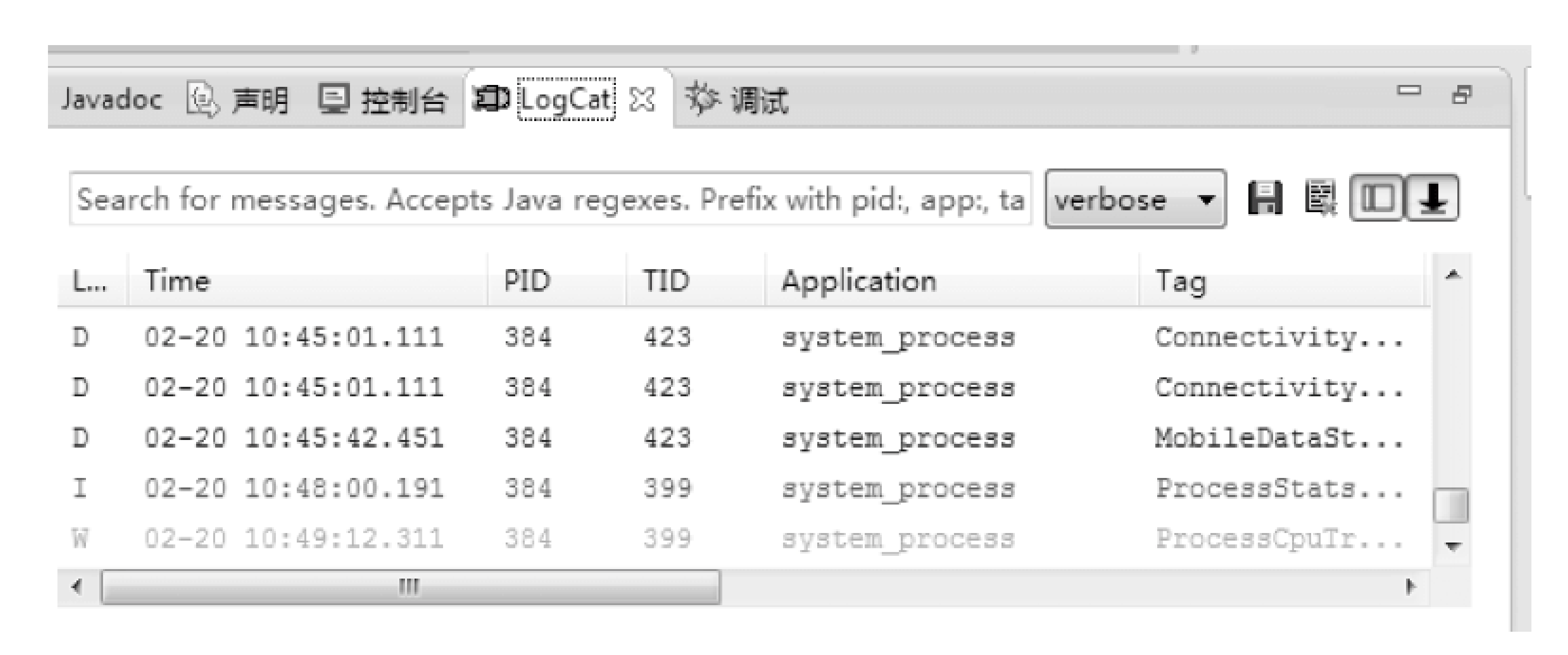


图 1-46 日志信息

在 LogCat 中显示所有测试终端操作的日志记录，通过不同颜色的输出可以明显区分警告信息和错误信息，并且可以使用右边的下拉菜单进行不同类型信息的筛选。

4. Debug 调试

由于 Android 应用程序使用 Java 语言编写，对 Android 应用程序的 Debug 调试和对标准 Java 语言的调试是相同的。

当在工程文件中标记了断点之后，可以使用下面两种方式开启调试。

(1) 右击项目，选择 Debug as，从应用程序开始运行就开始调试。

(2) 应用程序运行后，在 DDMS 界面的设备(Devices)选项卡中使用调试按钮进行调试。

第2章　Android 基本组件

Android 应用通常由一个或多个基本组件组成，本章对 Android 基本组件进行介绍。

2.1　Android 生命周期

程序同自然界中的生物一样，也有自己的生命周期。应用程序的生命周期即程序的存活时间。Android 为一构建在 Linux 之上的开源移动开发平台，在 Android 中，多数情况下每个程序都是在各自独立的 Linux 进程中运行的。当一个程序或其某些部分被请求时，它的进程就“出生”了，当这个程序没有必要再运行下去且系统需要回收这个进程的内存用于其他程序时，这个进程即“死亡”了。可以看出，Android 程序的生命周期是由系统控制而非程序自身直接控制。这和编写桌面应用程序时的思维有一些不同，一个桌面应用程序的进程也是在其他进程或用户请求时被创建，但往往是在程序自身收到关闭请求后执行一个特定的动作（比如从 main 函数中返回）而导致进程结束的。要想做好某种类型的程序或者某种平台下的程序的开发，最关键的是要弄清楚这种类型的程序或整个平台下的程序的一般工作模式并熟记在心。在 Android 中，程序的生命周期控制就属于这个范畴。

开发者必须了解不同的应用程序组件，尤其是 Activity、Service 和 Intent Receiver。了解这些组件是如何影响应用程序的生命周期的，这非常重要。如果不正确地使用这些组件，可能会导致系统终止正在执行重要任务的应用程序进程。

一个常见的进程生命周期漏洞的例子是 Intent Receiver（意图接收器），当 Intent Receiver 在 onReceive 方法中接收到一个 Intent（意图）时，其会启动一个线程，然后返回。一旦返回，系统将认为 Intent Receiver 不再处于活动状态，因而 Intent Receiver 所在的进程也就不再有用了（除非该进程中还有其他组件处于活动状态）。因此，系统可能会在任意时刻终止该进程以回收占用的内存，这样进程中创建出的那个线程也将被终止。解决这个问题的方法是从 Intent Receiver 中启动一个服务，让系统知道进程中还有活动状态的工作。为了使系统能够正确地决定在内存不足时应该终止进程，Android 根据每个进程中运行的组件及组件的状态把进程放入一个“Importance Hierarchy（重要性分组）”中。进程的类型按重要程度排列。

那么，一个 Android 程序的进程是何时被系统结束的呢？通俗地说，一个即将被系统关闭的程序是系统在内存不足（low memory）时根据“重要性层次”选出来的“牺牲品”。一个进程的重要性是根据其中运行的部件和部件的状态决定的。各种进程按照重要性从高到低

排列如图 2-1 所示。

(1) 前台进程：前台进程是与用户正在交互的进程，也是 Android 系统中最重要的进程。前台进程一般有以下 4 种情况：

① 进行中的 Activity 正在与用户进行交互。

② 进程服务被 Activity 调用，而且这个 Activity 正在与用户进行交互。

③ 进程服务正在执行生命周期中的回调函数，例如 onCreate()、onStart()或 onDestroy()。

④ 进程的 BroadcastReceiver 正在执行 onReceive()函数。

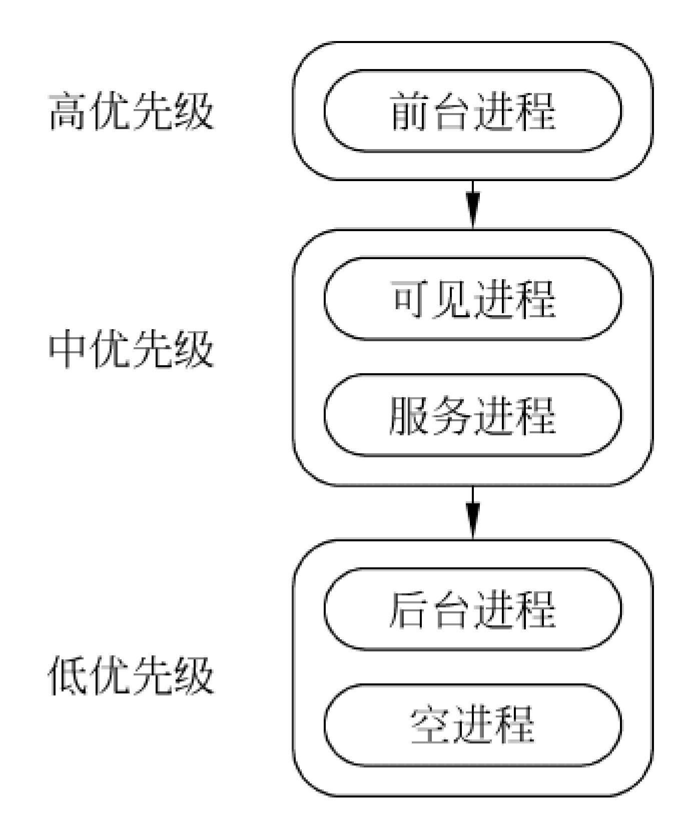


图 2-1　Android 进程的优先级

Android 系统为多任务操作系统，当系统中的多个前台进程同时运行时，如果出现资源不足的情况，Android 内核将自动清除部分前台进程，保证最主要的用户界面能够及时响应操作。

(2) 可见进程：在屏幕上显示，但是不在前台的程序。比如一个前台进程以对话框的形式显示在该进程前面。可见进程也很重要，它们只有在系统没有足够内存运行所有前台进程时才会结束。

(3) 服务进程：这样的进程在后台持续运行，比如后台音乐播放、后台数据的上传/下载等。这样的进程对用户来说一般很有用，所以只有当系统没有足够内存来维持所有的前台和可见进程时才会结束。

(4) 后台进程：这样的程序拥有一个用户不可见的 Activity，这样的程序在系统内存不足时按照 LRU 的顺序依次结束。

(5) 空进程：这样的进程不包含任何活动的程序部件，系统可能随时关闭这类进程。

从某种意义上讲，垃圾收集机制把程序员从"内存管理噩梦"中解放出来，而 Android 的进程生命周期管理机制把用户从"任务管理噩梦"中解放出来。Android 使用 Java 作为应用程序 API，并且结合其独特的生命周期管理机制为开发者和使用者提供最大程度的便利。

2.2　资源的管理与使用

第 1 章已经介绍了应用程序的目录组成结构，其中，res 目录表明了各种资源的存放位置，对于不同的资源，其存放的目录是不一样的，如表 2-1 所列。

表 2-1　资源存放目录表

目录结构	资源类型
res\anim\	XML 动画文件
res\drawable	位图文件
res\layout	XML 布局文件
res\values\	各种 XML 资源文件，主要包括以下几种。 array.xml：XML 数组文件 colors.xml：XML 颜色文件 dimens.xml：XML 尺寸文件 styles.xml：XML 风格文件
res\raw	原生文件

2.2.1 颜色资源

当需要在 Android 中使用不同颜色的时候，需要在 res\values 目录中新建 colors. xml 文件。在进行布局时，经常使用以下颜色：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < color name = "white"> # FFFFFF </color >                <! -- 白色 -- >
    < color name = "ivory"> # FFFFF0 </color >                <! -- 象牙色 -- >
    < color name = "lightyellow"> # FFFFE0 </color >          <! -- 亮黄色 -- >
    < color name = "yellow"> # FFFF00 </color >               <! -- 黄色 -- >
    < color name = "lavenderblush"> # FFF0F5 </color >        <! -- 淡紫色 -- >
    < color name = "papayawhip"> # FFEFD5 </color >           <! -- 象牙色 -- >
    < color name = "peachpuff"> # FFDAB9 </color >            <! -- 桃色 -- >
    < color name = "gold"> # FFD700 </color >                 <! -- 金色 -- >
    < color name = "pink"> # FFD700 </color >                 <! -- 粉红色 -- >
    < color name = "lightpink"> # FFC0CB </color >            <! -- 橙色 -- >
    < color name = "orange"> # FFA500 </color >               <! -- 亮粉红色 -- >
    < color name = "lightsalmon"> # FFA07A </color >          <! -- 亮肉色 -- >
    < color name = "darkorange"> # FF8C00 </color >           <! -- 暗橘黄色 -- >
    < color name = "coral"> # FF7F50 </color >                <! -- 珊瑚色 -- >
    < color name = "hotpink"> # FF69B4 </color >              <! -- 热粉红色 -- >
    < color name = "tomato"> # FF6347 </color >               <! -- 西红柿色 -- >
    < color name = "orangered"> # FF4500 </color >            <! -- 红橙色 -- >
    < color name = "deeppink"> # FFD700 </color >             <! -- 深粉红色  -- >
    < color name = "fuchsia"> # FF00FF </color >              <! -- 紫红色  -- >
    < color name = "magenta"> # FF00FE </color >              <! -- 红紫色  -- >
    < color name = "red"> # FF0000 </color >                  <! -- 红色  -- >
    < color name = "mintcream"> # F5FFFA </color >            <! -- 薄荷色  -- >
    < color name = "gainsboro"> # DCDCDC </color >            <! -- 淡灰色  -- >
    < color name = "crimson"> # DC143C </color >              <! -- 暗深红色  -- >
    < color name = "palevioletred"> # DB7093 </color >        <! -- 苍紫罗兰色  -- >
    < color name = "goldenrod"> # DAA520 </color >            <! -- 金麒麟色  -- >
    < color name = "orchid"> # DA70D6 </color >               <! -- 深紫色  -- >
    < color name = "silver"> # C0C0C0 </color >               <! -- 银色  -- >
    < color name = "darkkhaki"> # BDB76B </color >            <! -- 暗黄褐色  -- >
    < color name = "rosybrown"> # BC8F8F </color >            <! -- 褐玫瑰色  -- >
    < color name = "palegreen"> # 98FB98 </color >            <! -- 苍绿色  -- >
    < color name = "lightskyblue"> # 87CEEB </color >         <! -- 亮天蓝色  -- >
    < color name = "skyblue"> # 87CEEB </color >              <! -- 天蓝色  -- >
    < color name = "slategray"> # 708090 </color >            <! -- 灰石色  -- >
    < color name = "olivedrab"> # 6B8E23 </color >            <! -- 深绿褐色  -- >
    < color name = "slateblue"> # 6A5ACD </color >            <! -- 石蓝色  -- >
    < color name = "dimgray"> # 696969 </color >              <! -- 暗灰色 -- >
    < color name = "mediumaquamarine"> # 66CDAA </color >     <! -- 中绿色  -- >
    < color name = "cornflowerblue"> # 6495ED </color >       <! -- 菊蓝色  -- >
    < color name = "cadetblue"> # 5F9EA0 </color >            <! -- 军蓝色 -- >
    < color name = "darkolivegreen"> # 556B2F </color >       <! -- 暗橄榄绿色  -- >
    < color name = "indigo"> # 4B0082 </color >               <! -- 靛青色  -- >
    < color name = "mediumturquoise"> # 48D1CC </color >      <! -- 中绿宝石色  -- >
    < color name = "darkslateblue"> # 483D8B </color >        <! -- 暗灰蓝色  -- >
    < color name = "steelblue"> # 4682B4 </color >            <! -- 钢蓝色  -- >
    < color name = "royalblue"> # 4169E1 </color >            <! -- 皇家蓝色  -- >
```

```
    <color name="teal">#008080</color>                    <!-- 水鸭色   -->
    <color name="green">#008000</color>                   <!-- 绿色   -->
    <color name="darkgreen">#006400</color>               <!-- 暗绿色   -->
    <color name="blue">#0000FF</color>                    <!-- 蓝色   -->
    <color name="mediumblue">#0000CD</color>              <!-- 中蓝色   -->
    <color name="darkblue">#00008B</color>                <!-- 暗蓝色   -->
    <color name="navy">#000080</color>                    <!-- 海军色   -->
    <color name="black">#000000</color>                   <!-- 黑色   -->
</resources>
```

2.2.2 权限控制

在 Android 程序执行的过程中，在需要读取系统安全敏感项时，必须在 AndroidManifest.xml 中声明相关权限请求，也就是 Android 的访问权限。在开发过程中，不必把所有的权限都在 AndroidManifest.xml 中声明，只需选取所需权限声明即可，完整的权限列表如下。

- android.permission.ACCESS_CHECKIN_PROPERTIES：读取或写入登记 check-in 数据库属性表的权限。
- android.permission.ACCESS_COARSE_LOCATION：通过 WiFi 或移动基站的方式获取用户的经纬度信息，定位精度大概误差 30～1500m。
- android.permission.ACCESS_FINE_LOCATION：通过 GPS 芯片接收卫星的定位信息，定位精度在 10m 以内。
- android.permission.ACCESS_LOCATION_EXTRA_COMMANDS：允许程序访问额外的定位提供者指令。
- android.permission.ACCESS_MOCK_LOCATION：获取模拟定位信息，一般用于帮助开发者调试应用。
- android.permission.ACCESS_NETWORK_STATE：获取网络信息状态，例如当前的网络连接是否有效。
- android.permission.ACCESS_SURFACE_FLINGER：Android 平台上底层的图形显示支持，一般用于游戏或照相机预览界面和底层模式的屏幕截图。
- android.permission.ACCESS_WIFI_STATE：获取当前 WiFi 接入的状态以及 WLAN 热点的信息。
- android.permission.ACCOUNT_MANAGER：获取账户验证信息，主要为 GMail 账户信息，只有系统级进程才能访问的权限。
- android.permission.AUTHENTICATE_ACCOUNTS：允许一个程序通过账户验证方式访问账户管理 ACCOUNT_MANAGER 相关信息。
- android.permission.BATTERY_STATS：获取电池电量的统计信息。
- android.permission.BIND_APPWIDGET：允许一个程序告诉 AppWidget 服务需要访问小插件的数据库，只有非常少的应用才用到此权限。
- android.permission.BIND_DEVICE_ADMIN：请求系统管理员接收者 Receiver，只有系统才能使用。
- android.permission.BIND_INPUT_METHOD：请求 InputMethodService 服务，只有系统才能使用。

- android. permission. BIND_REMOTEVIEWS：必须通过 RemoteViewsService 服务来请求，只有系统才能使用。
- android. permission. BIND_WALLPAPER：必须通过 WallpaperService 服务来请求，只有系统才能使用。
- android. permission. BLUETOOTH：允许程序连接配过对的蓝牙设备。
- android. permission. BLUETOOTH_ADMIN：允许程序进行发现和配对新的蓝牙设备。
- android. permission. BRICK：能够禁用手机，非常危险，顾名思义就是“让手机变成砖头”。
- android. permission. BROADCAST_PACKAGE_REMOVED：当一个应用在删除时触发一个广播。
- android. permission. BROADCAST_SMS：当收到短信时触发一个广播。
- android. permission. BROADCAST_STICKY：允许一个程序收到广播后快速收到下一个广播。
- android. permission. BROADCAST_WAP_PUSH：WAP PUSH 服务收到后触发一个广播。
- android. permission. CALL_PHONE：允许程序从非系统拨号器里输入电话号码。
- android. permission. CALL_PRIVILEGED：允许程序拨打电话，替换系统的拨号器界面。
- android. permission. CAMERA：允许访问摄像头进行拍照。
- android. permission. CHANGE_COMPONENT_ENABLED_STATE：改变组件是否启用状态。
- android. permission. CHANGE_CONFIGURATION：允许当前应用改变配置，例如定位。
- android. permission. CHANGE_NETWORK_STATE：改变网络状态，例如是否能联网。
- android. permission. CHANGE_WIFI_MULTICAST_STATE，：改变 WiFi 多播状态。
- android. permission. CHANGE_WIFI_STATE：改变 WiFi 状态。
- android. permission. CLEAR_APP_CACHE：清除应用缓存。
- android. permission. CLEAR_APP_USER_DATA：清除应用的用户数据。
- android. permission. CWJ_GROUP：允许 CWJ 账户组访问底层信息。
- android. permission. CELL_PHONE_MASTER_EX：手机优化大师扩展权限。
- android. permission. CONTROL_LOCATION_UPDATES：允许获得移动网络定位信息改变。
- android. permission. DELETE_CACHE_FILES：允许应用删除缓存文件。
- android. permission. DELETE_PACKAGES：允许程序删除应用。
- android. permission. DEVICE_POWER：允许访问底层电源管理。
- android. permission. DIAGNOSTIC：允许程序 RW 诊断资源。

- android.permission.DISABLE_KEYGUARD：允许程序禁用键盘锁。
- android.permission.DUMP：允许程序从系统服务获取系统 dump 信息。
- android.permission.EXPAND_STATUS_BAR：允许程序扩展或收缩状态栏。
- android.permission.FACTORY_TEST：允许程序运行工厂测试模式。
- android.permission.FLASHLIGHT：允许访问闪光灯。
- android.permission.FORCE_BACK：允许程序强制使用 back 后退按键，无论 Activity 是否在顶层。
- android.permission.GET_ACCOUNTS：访问 GMail 账户列表。
- android.permission.GET_PACKAGE_SIZE：获取应用的文件大小。
- android.permission.GET_TASKS：允许程序获取当前或最近运行的应用。
- android.permission.GLOBAL_SEARCH：允许程序使用全局搜索功能。
- android.permission.HARDWARE_TEST：访问硬件辅助设备，用于硬件测试。
- android.permission.INJECT_EVENTS：允许访问本程序的底层事件，获取按键、轨迹球的事件流。
- android.permission.INSTALL_LOCATION_PROVIDER：安装定位提供。
- android.permission.INSTALL_PACKAGES：允许程序安装应用。
- android.permission.INTERNAL_SYSTEM_WINDOW：允许程序打开内部窗口，不对第三方应用程序开放此权限。
- android.permission.INTERNET：连接网络，可能产生 GPRS 流量。
- android.permission.KILL_BACKGROUND_PROCESSES：允许程序调用 killBackgroundProcesses(String)方法结束后台进程。
- android.permission.MANAGE_ACCOUNTS：允许程序管理 AccountManager 中的账户列表。
- android.permission.MANAGE_APP_TOKENS：管理创建、摧毁、Z 轴顺序，仅用于系统。
- android.permission.MTWEAK_USER：允许 mTweak 用户访问高级系统权限。
- android.permission.MTWEAK_FORUM：允许使用 mTweak 社区权限。
- android.permission.MASTER_CLEAR：允许程序执行软格式化，删除系统配置信息。
- android.permission.MODIFY_AUDIO_SETTINGS：修改声音设置信息。
- android.permission.MODIFY_PHONE_STATE：修改电话状态，如飞行模式，但不包含替换系统拨号器界面。
- android.permission.MOUNT_FORMAT_FILESYSTEMS：格式化可移动文件系统，比如格式化清空 SD 卡。
- android.permission.MOUNT_UNMOUNT_FILESYSTEMS：挂载、反挂载外部文件系统。
- android.permission.NFC：允许程序执行 NFC 近距离通信操作，用于移动支持。
- android.permission.PERSISTENT_ACTIVITY：创建一个永久的 Activity，该功能标记为将来将被移除。

- android. permission. PROCESS_OUTGOING_CALLS：允许程序监视，修改或放弃拨出的电话。
- android. permission. READ_CALENDAR：允许程序读取用户的日程信息。
- android. permission. READ_CONTACTS：允许应用访问联系人通讯录信息。
- android. permission. READ_FRAME_BUFFER：读取帧缓存用于屏幕截图。
- android. browser. permission. READ_HISTORY_BOOKMARKS：读取浏览器收藏夹和历史记录。
- android. permission. READ_INPUT_STATE：读取当前键的输入状态，仅用于系统。
- android. permission. READ_LOGS：读取系统底层日志。
- android. permission. READ_PHONE_STATE：访问电话状态。
- android. permission. READ_SMS：读取短信内容。
- android. permission. READ_SYNC_SETTINGS：读取同步设置，获得 Google 在线同步设置。
- android. permission. READ_SYNC_STATS：读取同步状态，获得 Google 在线同步状态。
- android. permission. REBOOT：允许程序重新启动设备。
- android. permission. RECEIVE_BOOT_COMPLETED：允许程序开机自动运行。
- android. permission. RECEIVE_MMS：接收彩信。
- android. permission. RECEIVE_SMS：接收短信。
- android. permission. RECEIVE_WAP_PUSH：接收 WAP PUSH 信息。
- android. permission. RECORD_AUDIO：通过手机或耳机的麦克录制声音。
- android. permission. REORDER_TASKS：重新排序系统 Z 轴运行中的任务。
- android. permission. RESTART_PACKAGES：通过 restartPackage(String)方法结束任务，该方式将在未来放弃。
- android. permission. SEND_SMS：发送短信。
- android. permission. SET_ACTIVITY_WATCHER：设置 Activity 观察器一般用于 monkey 测试。
- com. android. alarm. permission. SET_ALARM：设置闹铃提醒。
- android. permission. SET_ALWAYS_FINISH：设置程序在后台是否总是退出。
- android. permission. SET_ANIMATION_SCALE：设置全局动画缩放。
- android. permission. SET_DEBUG_APP：设置调试程序，一般用于开发。
- android. permission. SET_ORIENTATION：设置屏幕以横屏或标准方式显示，不用于普通应用。
- android. permission. SET_PREFERRED_APPLICATIONS：设置应用的参数已不再工作，具体查看 addPackageToPreferred(String)介绍。
- android. permission. SET_PROCESS_LIMIT：允许程序设置的最大进程数。
- android. permission. SET_TIME：设置系统时间。
- android. permission. SET_TIME_ZONE：设置系统时区。

- android. permission. SET_WALLPAPER：设置桌面壁纸。
- android. permission. SET_WALLPAPER_HINTS：设置壁纸建议。
- android. permission. SIGNAL_PERSISTENT_PROCESSES：发送一个永久的进程信号。
- android. permission. STATUS_BAR：允许程序打开、关闭、禁用状态栏。
- android. permission. SUBSCRIBED_FEEDS_READ：访问订阅信息的数据库。
- android. permission. SUBSCRIBED_FEEDS_WRITE：写入或修改订阅内容的数据库。
- android. permission. SYSTEM_ALERT_WINDOW：显示系统窗口。
- android. permission. UPDATE_DEVICE_STATS：更新设备状态。
- android. permission. USE_CREDENTIALS：允许程序从 AccountManager 请求验证。
- android. permission. USE_SIP：允许程序使用 SIP 视频服务。
- android. permission. VIBRATE：允许振动。
- android. permission. WAKE_LOCK：允许程序在手机屏幕关闭后后台进程仍然运行。
- android. permission. WRITE_APN_SETTINGS：写入网络 GPRS 接入点设置。
- android. permission. WRITE_CALENDAR：写入日程，但不可读取。
- android. permission. WRITE_CONTACTS：写入联系人，但不可读取。
- android. permission. WRITE_EXTERNAL_STORAGE：允许程序写入外部存储，例如在 SD 卡上写文件。
- android. permission. WRITE_GSERVICES：允许程序写入 Google Map 服务数据。
- com. android. browser. permission. WRITE_HISTORY_BOOKMARKS：写入浏览器历史记录或收藏夹，但不可读取。
- android. permission. WRITE_SECURE_SETTINGS：允许程序读/写系统安全敏感的设置项。
- android. permission. WRITE_SETTINGS：允许读/写系统设置项。
- android. permission. WRITE_SMS：允许编写短信。
- android. permission. WRITE_SYNC_SETTINGS：写入 Google 在线同步设置。

在程序的实际开发过程中，需要谨慎地选择权限声明，下面来动手声明权限。例如要声明一个网络权限，以便让程序联网，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.helloandroid"
    android:versionCode = "1"
    android:versionName = "1.0" >
< uses - permission android:name = "android.permission.INTERNET"></uses_permission >
</manifest >
```

其中，

- <uses_permission>：Android 系统安全敏感项权限声明标签。

- android:name：<uses_permission>的标签元素，指定 Android 系统安全敏感项的名称。
- android.permission.INTERNET：允许程序打开网络套接字。

2.3 Activity

在 Android 应用中可以有多个 Activity（活动），这些 Activity 组成了 Activity 栈（Stack），活动的 Activity 位于栈顶，之前的 Activity 被压入下面，成为非活动 Activity，等待是否可能被恢复为活动状态。

Activity 是最基本的 Android 应用程序组件，在应用程序中，一个活动通常就是一个单独的屏幕。每一个活动都被实现为一个独立类，并且从活动基类中继承而来，活动类将会显示由视图控件组成的用户接口，并对事件做出响应。在 Android 应用中，可以有一个或多个活动。

2.3.1 单个 Activity

1. Activity 的生命周期

Activity 的生命周期指应用程序的 Activity 从启动到销毁的全过程。Activity 的生命周期是在 Android 应用程序设计中最重要的内容，直接关系到用户程序的界面和功能。

每一个活动的状态都是由它在活动栈中处的位置决定的，活动栈是当前所有正在运行的进程的后进先出的集合。当一个新的活动启动时，当前的前台屏幕就会移动到栈顶。如果用户使用 Back（返回）按钮返回到了刚才的活动，或者前台活动被关闭了，那么栈中的下一个活动就会移动到栈顶，变为活动状态。图 2-2 说明了这个过程。

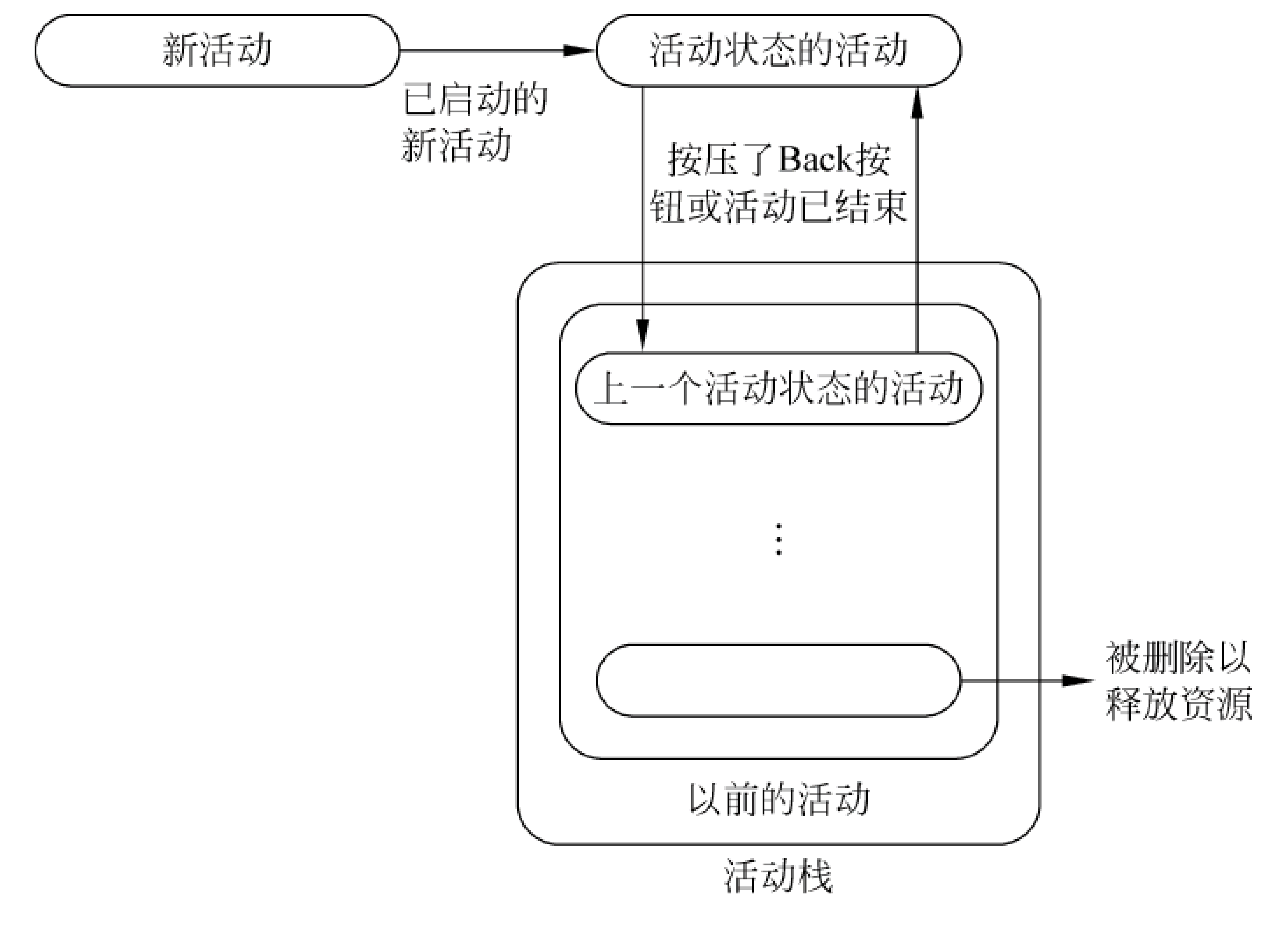


图 2-2 活动栈流程图

2. Activity 的状态

随着活动的创建和销毁，它们会按照图 2-2 所示的那样从栈中移进移出。在这个过程中，它们也经历了活动、暂停、停止和非活动 4 种可能的状态，如图 2-3 所示。

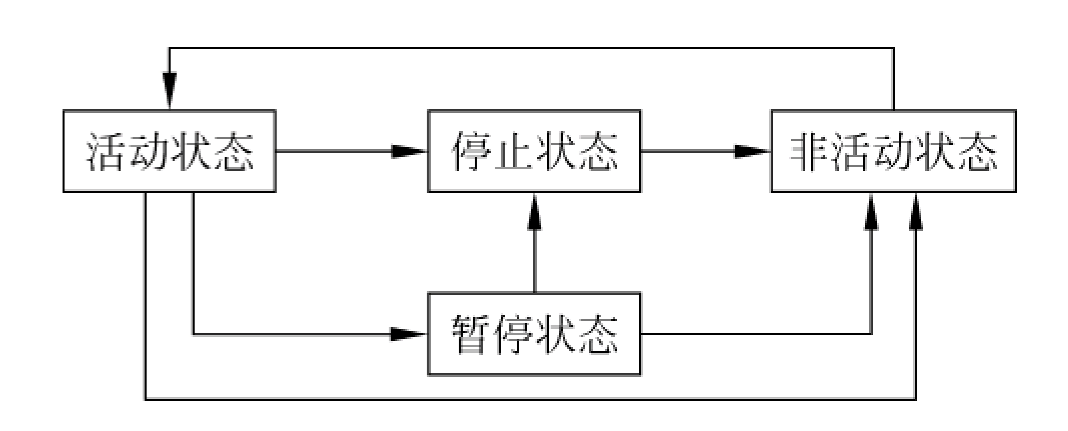


图 2-3　Activity 的状态

(1) 活动状态：当一个活动位于栈顶的时候，它是可见的、被关注的前台活动，这时它可以接收用户输入。Android 将会不惜一切代价保持它处于活动状态，并根据需要销毁栈下面部分的活动，以保证这个活动拥有它所需要的资源。当另一个活动变为活动状态时，这个活动将被暂停。

(2) 暂停状态：在某些情况下，活动是可见的，但是没有被关注，此时它处于暂停状态。当一个透明的或者非全屏的活动位于某个处于活动状态的活动之前时，这个透明的或者非全屏的活动就会达到这个状态。当活动被暂停的时候，它仍然会被当作近似于活动状态的状态，但是它不能接收用户的输入事件。在极端情况下，当一个活动变得完全不可见的时候，它就会变为停止状态。

(3) 停止状态：当一个活动不可见的时候，它就处于停止状态。此时，活动仍然会停留在内存中，保存所有的状态和成员信息，然而当系统的其他地方要求使用内存的时候，它们就会成为被清除的首要候选对象。在一个活动停止的时候，保存数据和当前的 UI 状态是很重要的。一旦一个活动被退出或者关闭，它就会变为非活动状态。

(4) 非活动状态：当一个活动被销毁之后，在它启动之前处于非活动状态。处于非活动状态的活动已经从活动栈中移除了，因此，在它们可以被显示和使用之前需要被重新启动。

在 Android 系统中，采用"栈"结构来管理 Activity，这是一种"后进先出"的原则，如图 2-4 所示。当一个 Activity 被启用时，将执行入栈操作。位于栈顶的 Activity 处于活动状态，其他 Activity 处于暂停状态或者停止状态。当 Activity 关闭时，将执行出栈操作，从而变成非活动状态。当 Android 系统资源紧张时，Android 内核会终止部分长久没有响应的 Activity，使之成为非活动状态，从而释放系统资源。

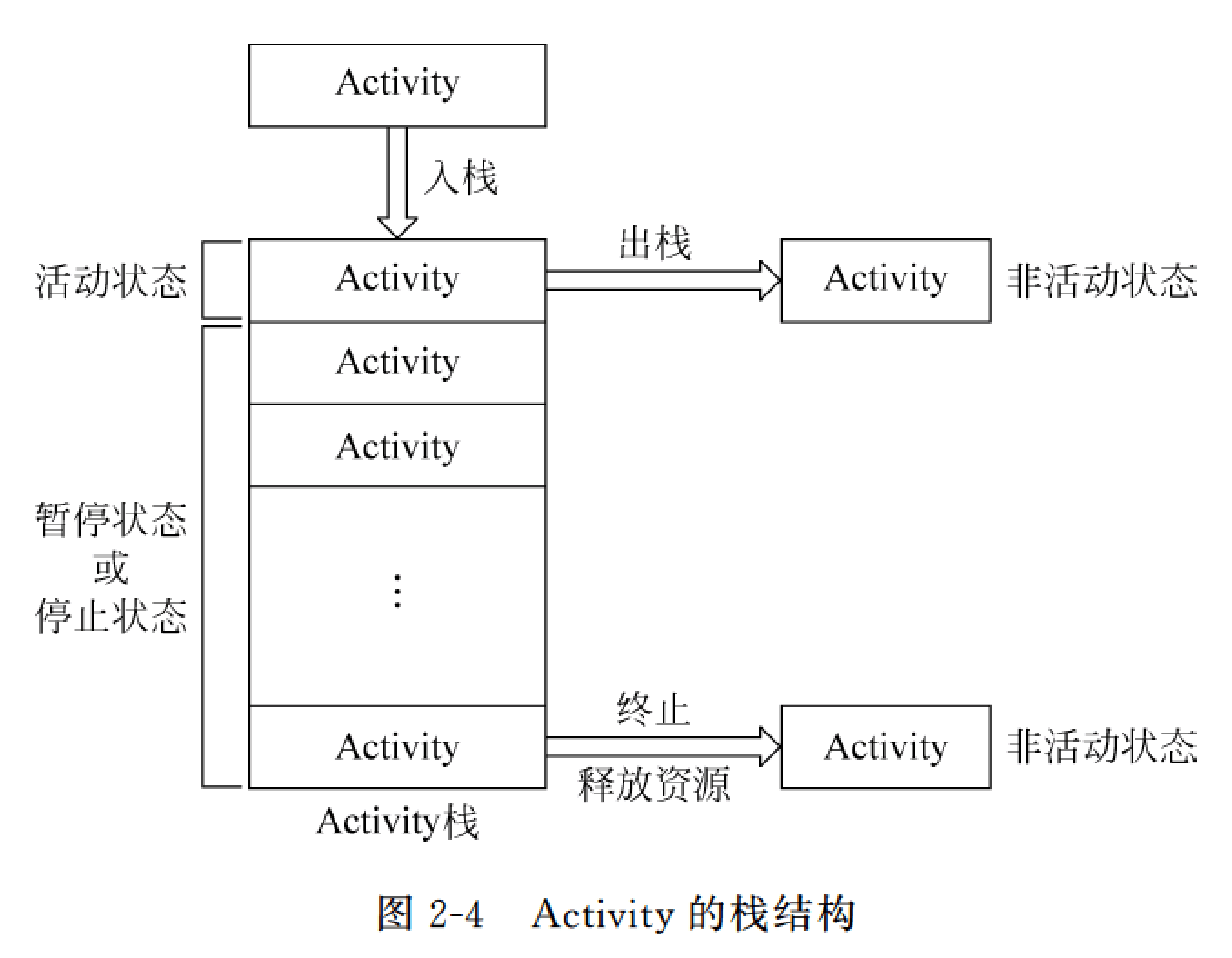


图 2-4　Activity 的栈结构

3. Activity 生命周期的管理

在 Activity 系统中，一般通过 Activity 的事件回调函数来管理 Activity 的生命周期。这些事件回调函数如下：

```
public class MyActivity extends Activity
{
void onCreate(Bundle savedInstanceState);
void onStart();void onRestart();
    void onResume();
    void onPause();
    void onStop();
    void onDestroy();
}
```

这些 Activity 生命周期的事件回调函数将会被 Android 系统自动调用，用户也可以重载这些事件回调函数来完成自己的操作。Activity 生命周期的事件回调函数的说明如表 2-2 所列。

表 2-2　Activity 生命周期的事件回调函数

函　　数	是否可终止	描　　述
onCreate()	否	Activity 启动后第一个被调用的函数，常用来进行 Activity 的初始化，例如创建 View、绑定数据或恢复信息等
onStrt()	否	当 Activity 显示在屏幕上时，该函数被调用
onRestart()	否	当 Activity 从停止状态进入活动状态时，调用该函数
onResume()	否	当 Activity 能够与用户交互接受用户输入时，该函数被调用。此时的 Activity 位于栈顶
onPause()	是	当 Activity 进入暂停状态时，该函数被调用，一般用来保存持久的数据或释放占用的资源
onStop()	是	当 Activity 进入停止状态时，该函数被调用
onDestroy()	是	在 Activity 被终止时，即进入非活动状态前，该函数被调用

在 Android 系统中，Activity 的生命周期以及各个事件回调函数之间的跳转如图 2-5 所示。

另外，当系统资源紧张时，Activity 可能会被终止。此时，Android 提供了相应的状态保存/恢复的事件回调函数，如表 2-3 所列。

表 2-3　Activity 状态保存/恢复的事件回调函数

函数	是否可终止	描　　述
onSaveInstanceState()	否	Android 系统因资源不足终止 Activity 前调用该函数，用于保存 Activity 的状态，供 onRestoreInstanceState()或 onCreate()恢复用
onRestoreInstanceState()	否	恢复 onSaveInstanceState()保存的 Activity 状态信息，在 onStart()和 onResume()之间被调用

4. Activity 的启动

在一个 Android 项目中，如果只有一个 Activity，那么只需要在 AndroidManifest.xml 文件中对其进行配置，并且将其设置为程序的入口。这样，当运行该项目时，将自动启动该

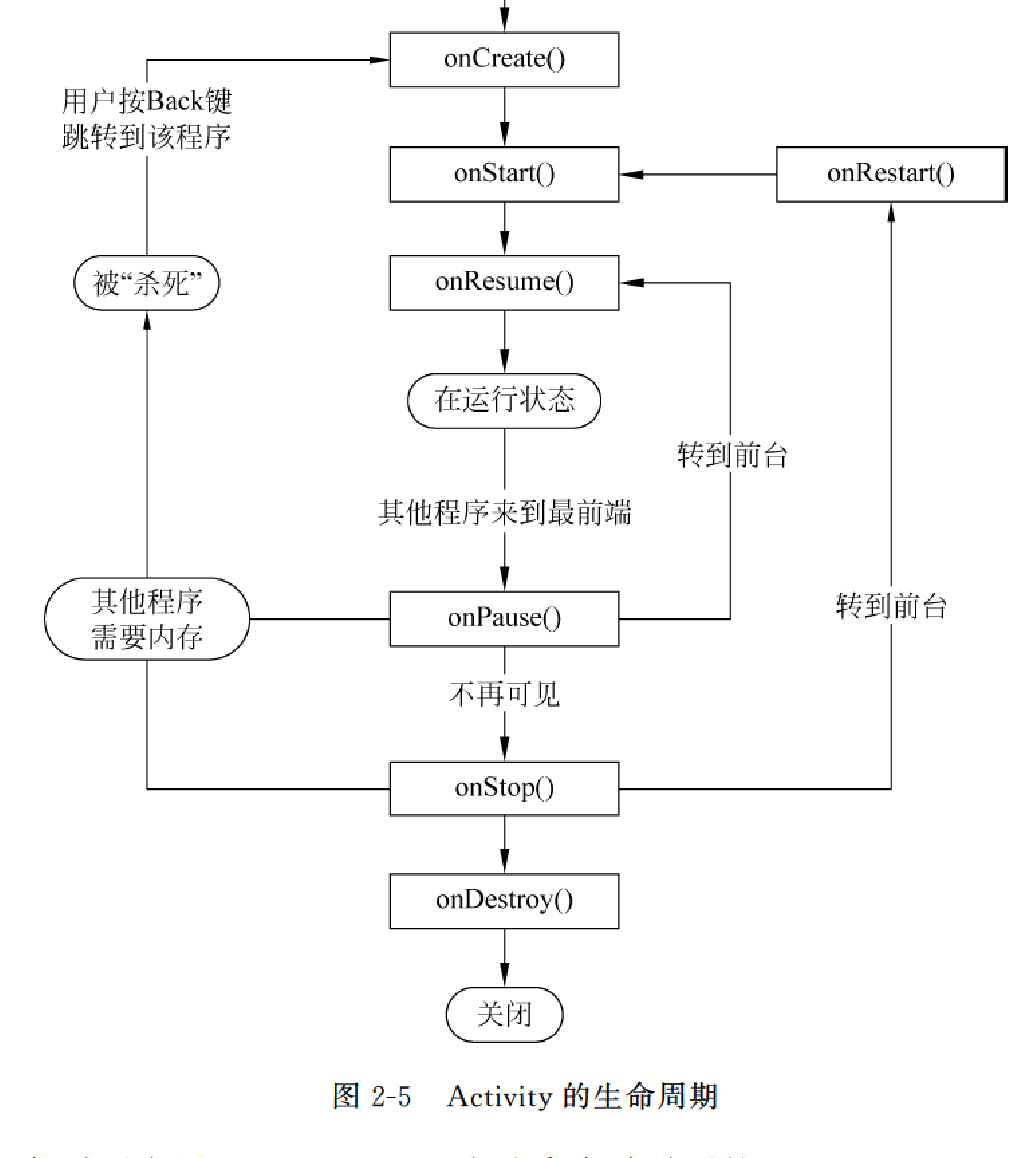


图 2-5　Activity 的生命周期

Activity。否则，需要应用 startActivity()方法来启动需要的 Activity。startActivity()方法的格式为：

```
public void startActivity(Intent intent)
```

该方法没有返回值，只有一个 Intent 类型的入口参数，Intent 是 Android 应用中各组件之间的通信方式，一个 Activity 通过 Intent 来表达自己的“意图”。在创建 Intent 对象时，需要指定想要被启动的 Activity。

例如要启动一个名为 runActivity 的 Activity，代码为：

```
Intent intent = new Intent(MainActivity.this,runActivity.class);
startActivity(intent);
```

5. Activity 的关闭

在 Android 中，如果想关闭当前的 Activity，可使用 Activity 类提供的 finish()方法。finish 方法的格式为：

```
public void finish()
```

该方法的使用比较简单，既没有入口参数，也没有返回值，只需要在 Activity 相应的事件中调用该方法即可。例如，想要在单击按钮时关闭 Activity，代码为：

```
Button button1 = (Button)findViewByld(R.id.button1);
button1.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v){
        finish();                                    //关闭当前 Activity
    }
})
```

注意：如果当前的 Activity 不是主活动，那么执行 finish()方法后，将返回到调用它的那个 Activity；否则，将返回到主屏幕中。

6. Activity 生命周期的调用顺序

一般来说，Activity 的生命周期可分为全生命周期、可视生命周期和活动生命周期三类。每种生命周期中包含不同的事件回调函数，各个事件回调函数的调用顺序也不同。

对于 Activity 全生命周期，事件回调函数的调用顺序为 onCreate()→onStart()→onResume()→onPause()→onStop()→onDestroy()。每一步调用的含义如下：

- 调用 onCreate()函数分配资源。
- 调用 onStart()将 Activity 显示在屏幕上。
- 调用 onResume()获取屏幕焦点。
- 调用 onPause()、onStop 和 onDestroy()，释放资源并销毁进程。

对于 Activity 可视生命周期，事件回调函数的调用顺序为 onSaveInstanceState()→onPause()→onStop()→onRestart()→onStart()→onResume()。每一步调用的含义如下：

- 调用 onSaveInstanceState()函数保存 Activity 状态。
- 调用 onPause()和 onStop()，停止对不可见 Activity 的更新。
- 调用 onRestart()恢复界面上需要更新的信息。
- 调用 onStart()和 onResume()重新显示 Activity，并接受用户交互。

对于活动生命周期，事件回调函数的调用顺序为 onSaveInstanceState()→onPause()→onResume()。每一步调用的含义如下：

- 调用 onSaveInstanceState()保存 Activity 的状态。
- 调用 onPause()停止与用户交互。
- 调用 onResume()恢复与用户的交互。

7. 应用实例

下面通过一个实例来说明单个 Activity 的应用。

【例 2-1】 实现应用对话框主题的 Activity 实例。

其实现步骤如下：

(1) 在 Eclipse 中建立 Android 应用文件，命名为 li2_1Activity。

(2) 修改新建项目的 res\layout 目录下的布局文件 main.xml，在布局文件中应用线性布局完成一个按钮的游戏开始界面。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
```

```
<!-- 添加顶部图片 -->
<ImageView android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:scaleType = "centerCrop"
    android:layout_weight = "1.2"
    android:src = "@drawable/top" />
<!-- 添加一个相对布局管理器 -->
<RelativeLayout android:layout_weight = "2"
    android:layout_height = "wrap_content"
    android:background = "@drawable/bottom"
    android:id = "@ + id/relativeLayout1"
    android:layout_width = "match_parent">
    <!-- 添中间位置的图片 -->
    <ImageView android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/imageButton0"
        android:src = "@drawable/in"
        android:layout_alignTop = "@ + id/imageButton5"
        android:layout_centerInParent = "true" />
    <!-- 添加上显示的图片 -->
    <ImageView android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/imageButton1"
        android:src = "@drawable/setting"
        android:layout_above = "@ + id/imageButton0"
        android:layout_alignRight = "@ + id/imageButton0" />
    <!-- 添加下方显示的图片 -->
    <ImageView android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/imageButton2"
        android:src = "@drawable/exit"
        android:layout_below = "@ + id/imageButton0"
        android:layout_alignLeft = "@ + id/imageButton0" />
    <!-- 添加左侧显示的图片 -->
    <ImageView android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/imageButton3"
        android:src = "@drawable/help"
        android:layout_toLeftOf = "@ + id/imageButton0"
        android:layout_alignTop = "@ + id/imageButton0" />
    <!-- 添加右侧显示的图片 -->
    <ImageView android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/about"
        android:src = "@drawable/about"
        android:layout_toRightOf = "@ + id/imageButton0"
        android:layout_alignTop = "@ + id/imageButton0" />
</RelativeLayout>
</LinearLayout>
```

(3) 创建一个继承类，命名为 Aboutgame.java。其具体操作为：

首先选择 src\fs.intent1 文件夹右击，在弹出的快捷菜单中选择“新建”→“类”命令，弹出“新建 Java 类”对话框，如图 2-6 所示。然后在“名称”右侧的文本框中输入对应的类名，

单击“超类”右侧的“浏览”按钮，弹出“选择超类”对话框(如图 2-7 所示)，在“选择类型”下面的文本框中输入 Activity，即弹出对应的选择，选择 android. app. Activity 可定义 Activity 类，至此完成新类的创建。



图 2-6 “新建 Java 类”对话框

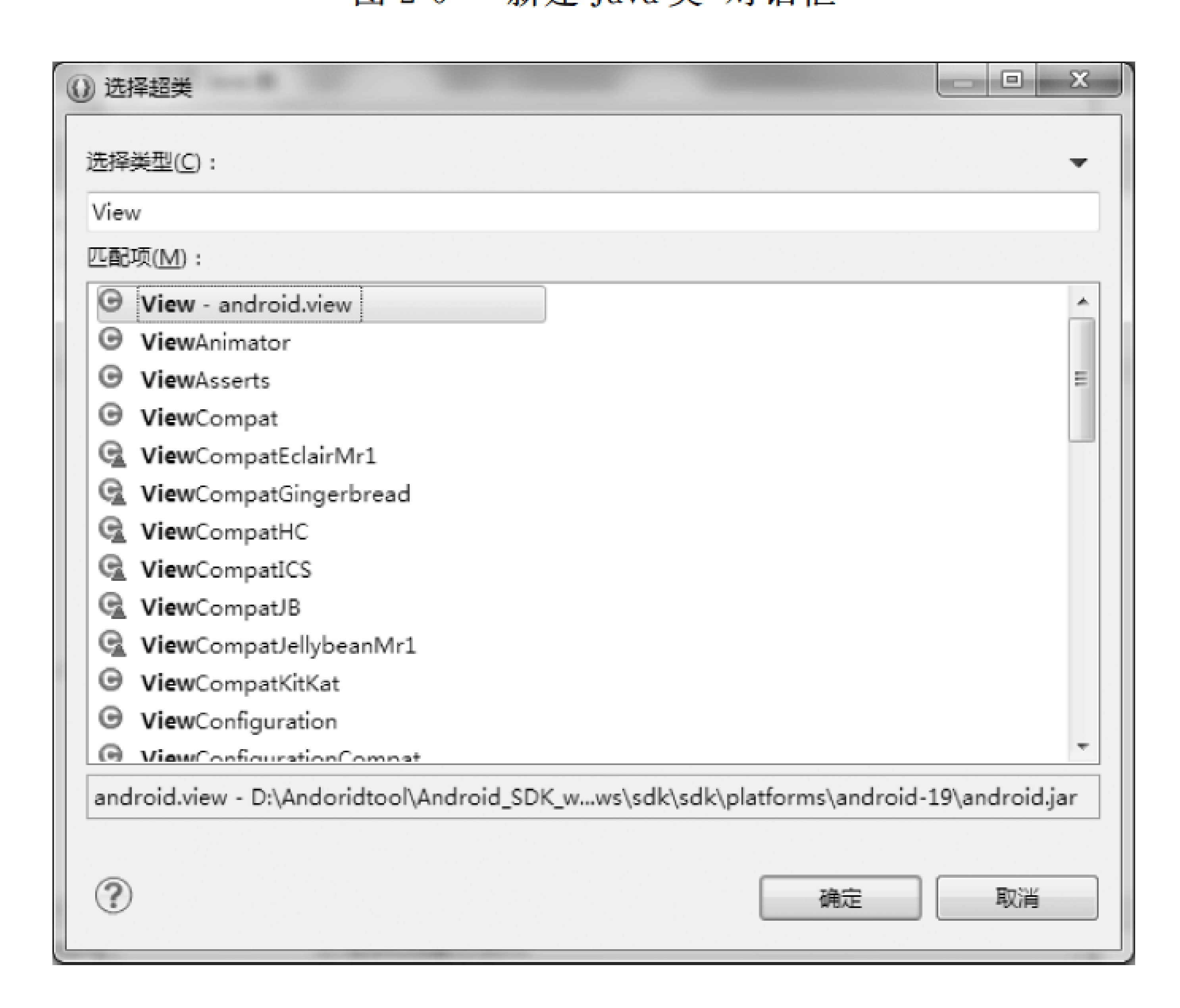


图 2-7 “选择超类”对话框

该类用于重写 onCreate()方法，在重写的 onCreate()方法中，先创建一个线性布局管理器对象，并设置其内边距，然后创建一个 TextView 对象，并设置字体大小及要显示的内容，再将 TextView 添加到线性布局管理器中，最后设置在该 Activity 中显示线性布局管理器对象：

```
public class Aboutgame extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        LinearLayout ll = new LinearLayout(this);      //创建线性布局管理器对象
        ll.setPadding(20,20,20,20);
        TextView tv = new TextView(this);              //创建 TextView 对象
        tv.setTextSize(24);                            //设置字体大小
        tv.setText(R.string.about);                    //设置要显示的内容
        ll.addView(tv);                                //将 TextView 添加到线性布局管理器中
        setContentView(ll);                            //设置该 Activity 显示的内容视图
    }
}
```

(4) 在 MainActivity 代码中为 TextView 控件设置要显示的文本内容时，采用的是使用字符串资源的方法。打开 res\values 目录下的 strings.xml 文件，在文件中添加一个名称为 about 的字符串变量，内容是关于要显示的信息，将代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name">Activity 关于对话框主题</string>
    <string name = "action_settings">Settings</string>
<string name = "about">泡泡龙游戏是一款十分流行的益智游戏.它可以从下方中央的弹珠发射台射
出彩珠,当有多于 3 个同色弹珠相连时,这些弹珠将会爆掉,否则该弹珠被连接到指向的位置,直到泡
泡下压越过下方的警戒线时游戏结束。</string>
</resources>
```

(5) 修改创建的主活动 MainActivity，在该文件的 onCreate()方法中获取“关于”按钮并为其添加单击事件监听器，在重写的 onClick()方法中创建一个 Aboutgame 所对应的 Intent 对象，并调用 startActivity()方法，启动 Aboutgame：

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView about = (ImageView)findViewById(R.id.about);    //获取“关于”按钮
        about.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                //创建 Intent 对象
                Intent intent = new Intent(MainActivity.this, Aboutgame.class);
                startActivity(intent);                 //启动关于 Activity
            }
        });
    }
}
```

(6) 在 AndroidManifest.xml 文件中配置 Aboutgame，配置的主要属性有 Activity 使用的图标、实现类、标签和使用的主题。代码为：

```
<Activity
    android:name = ".MainActivity"
    android:label = "@string/app_name" >
    <intent - filter>
        <action android:name = "android.intent.action.MAIN" />
        <category android:name = "android.intent.category.LAUNCHER" />
    </intent - filter>
</Activity>
        <Activity
     android:icon = "@drawable/ic_launcher"
     android:name = ".Aboutgame"
     android:label = "关于..."
     android:theme = "@android:style/Theme.Dialog">
        </Activity>
    </application>
</manifest>
```

运行程序，显示泡泡龙游戏主界面，如图 2-8(a)所示，单击“关于”按钮，将显示如图 2-8(b)所示的对话框。




(a) 游戏主界面



(b) “关于”对话框

图 2-8　对话框主题效果

2.3.2　多个 Activity

在 Android 应用中，经常会遇到多个 Activity，而这些 Activity 之间又经常需要交换数据。

1. Activity 间的数据交换

当在一个 Activity 中启动另一个 Activity 时，经常需要传递一些数据，这时可以通过 Intent 来实现。因为 Intent 通常被称为两个 Activity 之间的信使，通过将要传递的数据保存在 Intent 中，就可以将其传递到另一个 Activity 中了。

在 Android 中，可以将要保存的数据存放在 Bundle 对象中，然后通过 Intent 提供的 putExtras()方法将要携带的数据保存到 Intent 中。

2. 调用另一个 Activity

在开发 Android 应用时，有时需要在一个 Activity 中调用另一个 Activity，当用户在第 2 个 Activity 中选择完成后，程序会自动返回到第 1 个 Activity 中，第 1 个 Activity 必须能够获取并显示用户在第 2 个 Activity 中选择的结果；或者，在第 1 个 Activity 中将一些数据传递到第 2 个 Activity，由于某些原因，又要返回到第 1 个 Activity 中，并显示传递的数据，例如程序中经常出现的“返回上一步”功能，这时也可以通过 Intent 和 Bundle 来实现。与在两个 Activity 之间交换数据不同的是，此处需要使用 startActivityForResult()方法来启动另一个 Activity。

3. 应用实例

下面通过一个实例来说明在 Android 中多个 Activity 的应用。

【例 2-2】 利用多个 Activity 实现根据身高计算标准体重。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li2_2Activity。

(2) 选择 res\layout 目录下的布局文件 main. xml，将默认的相对布局管理器改为垂直线性布局管理器，并添加用于选择性别信息的单选按钮组、用于输入身高的文本框以及一个 OK 按钮：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "@drawable/bj1" >
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center_horizontal"
        android:padding = "20px"
        android:text = "标准体重" />
    <LinearLayout
        android:id = "@ + id/linearLayout1"
        android:gravity = "center_vertical"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" >
        <TextView
            android:id = "@ + id/textView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "性别: " />
        <RadioGroup
```

```
            android:id = "@ + id/sex"
            android:orientation = "horizontal"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content" >
            < RadioButton
                android:id = "@ + id/radio0"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:checked = "true"
                android:text = "男" />
            < RadioButton
                android:id = "@ + id/radio1"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "女" />
        </RadioGroup >
    </LinearLayout >
    < LinearLayout
        android:id = "@ + id/linearLayout1"
        android:gravity = "center_vertical"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" >
        < TextView
            android:id = "@ + id/textView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "身高: " />
        < EditText
            android:id = "@ + id/stature"
            android:minWidth = "100px"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content">
        </EditText >
        < TextView
            android:id = "@ + id/textView2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "cm" />
    </LinearLayout >
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "OK" />
</LinearLayout >
```

(3) 选择 res\layout 目录,然后右击,在弹出的快捷菜单中选择“新建→文件”命令,弹出如图 2-9 所示的“新建文件”对话框,在“文件名”右侧的文本框中输入 upshot. xml,单击“确定”按钮,完成布局文件的创建。

在该布局文件中采用垂直线性布局管理器,并且添加 3 个 TextView 控件,分别用于显示性别、身高和计算后的标准体重:



图 2-9 “新建文件”对话框

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical"
    android:background = "@drawable/bj1">
    < TextView
        android:id = "@ + id/sex"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:padding = "10px"
        android:text = "性别" />
    < TextView
        android:id = "@ + id/stature"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:padding = "10px"
        android:text = "身高" />
    < TextView
        android:id = "@ + id/weight"
        android:padding = "10px"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "标准体重" />
</LinearLayout >
```

(4) 新建一个实现 java.io.Serializable 接口的 Java 类，并将其命名为 SexJ.java：

```
public class SexJ implements Serializable {
    private static final long serialVersionUID = 1L;
    private String sex = "";                          //性别
    private int stature = 0;                          //身高
    public String getSex() {                          //添加属性方法
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
    public int getStature() {
        return stature;
    }
    public void setStature(int stature) {
        this.stature = stature;
    }
}
```

(5) 打开 res\layout 目录下的 MainActivity，在 onCreate()方法中获取 OK 按钮，并为其添加单击事件监听器。在重写的 onClick()方法中，实例化一个保存性别和身高的可序列化对象 SexJ，并判断输入的身高是否为空，如果为空，给出消息提示并返回；否则，首先获取性别和身高并保存到 SexJ 中，然后实例化一个 Bundle 对象，并将输入的身高和性别保存到 Bundle 对象中，接着创建一个启动显示结果 Activity 的 Intent 对象，并将 Bundle 对象保存到该 Intent 对象中，最后启动 Intent 对应的 Activity。代码为：

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button)findViewById(R.id.button1);
        button.setOnClickListener(new OnClickListener(){
    @Override
    public void onClick(View v){
    SexJ sexj = new SexJ();                           //实例化一个保存输入基本信息的对象
    if("".equals(((EditText)findViewById(R.id.stature)).getText().toString())){
    Toast.makeText(MainActivity.this, "请输入身高,否则不能计算!", Toast.LENGTH_SHORT).
show();
    return;
    }
     int stature = Integer.parseInt(((EditText)findViewById(R.id.stature)).getText().
toString());
   //获取设置性别的单选按钮组
    RadioGroup sex = (RadioGroup)findViewById(R.id.sex);
    //获取单选按钮组的值
    for(int i = 0;i < sex.getChildCount();i++){
    RadioButton r = (RadioButton)sex.getChildAt(i);   //根据索引值获取单选按钮
    if(r.isChecked()){                                //判断单选按钮是否被选中
    sexj.setSex(r.getText().toString());              //获取被选中的单选按钮的值
    break;
```

```
                //跳出 for 循环
                }
                 }
                 sexj.setStature(stature);                          //设置身高
                 Bundle bundle = new Bundle();                      //实例化一个 Bundle 对象
                 bundle.putSerializable("sexj", sexj);              //将输入的基本信息保存到 Bundle 对象中
                 Intent intent = new Intent(MainActivity.this,upshotActivity.class);
                 intent.putExtras(bundle);                          //将 bundle 保存到 Intent 对象中
                 startActivity(intent);                             //启动 Intent 对应的 Activity
                 }
            });
            }
        }
```

(6) 创建一个继承 Activity 类,命名为 upshotActivity,重写 onCreate()方法。在重写的 onCreate()方法中,先设置该 Activity 使用的布局文件 upshot.xml 中定义的布局,接着获取性别、身高和体重文本框,再获取 Intent 对象以及传递的数据包,最后将传递过来的性别、身高和计算后的体重显示到对应的文件框中。

```
public class upshotActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.upshot);                    //设置该 Activity 使用的布局
        TextView sex = (TextView)findViewById(R.id.sex);            //获取显示性别的文本框
        //获取显示身高的文本框
        TextView stature = (TextView)findViewById(R.id.stature);
        //获取显示标准体重的文本框
        TextView weight = (TextView)findViewById(R.id.weight);
        Intent intent = getIntent();                        //获取 Intent 对象
        Bundle bundle = intent.getExtras();                 //获取传递的数据包
        //获取一个可序列化的 Info 对象
        SexJ sexj = (SexJ)bundle.getSerializable("sexj");
        sex.setText("您是一位" + sexj.getSex() + "士");  //获取性别并显示到相应文本框中
        //获取身高并显示到相应文本框中
        stature.setText("身高为" + sexj.getStature() + "厘米");
        //显示计算后的标准体重
    weight.setText("标准体重为" + getWeight(sexj.getSex(),sexj.getStature()) + "公斤");
    }
    private String getWeight(String sex,float stature){
        String weight = "";                                 //保存体重
        NumberFormat format = new DecimalFormat();
        if(sex.equals("男")){                               //计算男士标准体重
            weight = format.format((stature - 80) * 0.7);
        }else{                                              //计算女士标准体重
            weight = format.format((stature - 70) * 0.6);
        }
        return weight;
    }
}
```

(7) 在 AndroidManifest.xml 文件中配置 upshotActivity,配置的主要属性有 Activity 使用标签、图标和实现类。

```
        ⋮
        <Activity
            android:name = "fs.li2_2Activity.MainActivity"
            android:label = "@string/app_name" >
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </Activity>
        <Activity
            android:label = "结果"
            android:icon = "@drawable/ic_launcher"
            android:name = ".upshotActivity">
        </Activity>
    </application>
</manifest>
```

运行程序,将显示一个输入计算标准体重条件的界面,选择性别并输入身高,如图 2-10 所示,然后单击 OK 按钮,效果如图 2-11 所示。





图 2-10　主界面



图 2-11　显示计算结果

2.4　Intent

一个 Android 程序由多个组件组成,各个组件之间使用 Intent(意图)进行通信。Intent 对象中包含组件名称、动作、数据等内容。根据 Intent 中的内容,Android 系统可以启动需要的组件。

Intent 有显式和隐式之分,显式的 Intent 是根据组件的名称直接启动要启动的组件,例如 Service 或者 Activity;隐式的 Intent 通过配置的 Action、Category、Data 来找到匹配的

组件并启动。

2.4.1 Intent 构成与属性

要在不同的 Activity 之间传递数据，就要在 Intent 中包含相应的东西。一般来说，在数据中最起码包括以下两点。

- Action：用来指明要实施的动作是什么，例如 ACTION _ VIEW、ACTION _ EDIT 等。
- Data：要实施的具体数据一般由一个 URI 变量来表示。例如：
  ACTION_VIEW content://contacts/1：显示 identifier 为 1 的联系人的信息。
  ACTION_VIEW content://contacts/1：给该联系人拨打电话。

下面详细介绍 Intent 的各属性值，以及 Android 怎样根据不同的属性值来启动相应的组件。

1. Component 属性

Intent 的 Component 属性需要接受一个 ComponentName 对象，ComponentName 的语法格式如下。

- ComponentName(String pkg,String cls)：创建 pkg 所在包下的 cls 对应的组件。
- ComponentName(Context pkg,String cls)：创建 pkg 所在包下的 cls 类对应的组件。
- ComponentName(Context pkg,Class<? >cls)：创建 pkg 所在包下的 cls 类对应的组件。

上面的语法格式本质上为一个，说明创建一个 ComponentName 需要指定包名和类名——这就可唯一地确定一个组件类，这样应用程序即可根据给定的组件类去启动特定的组件。

2. Action、Category 属性

Intent 的 Action、Category 属性都是一个普通的字符串，其中 Action 代表该 Intent 所要完成的一个抽象“动作”，而 Category 用于为 Activity 增加额外的附加类别信息。通常，Action 属性会和 Category 属性结合使用。

Action 要完成的只是一个抽象的动作，这个动作具体用哪个组件(或者是 Activity，或者是 BroadcastReceiver)来完成，Action 这个字符串本身并不管。例如，Android 提供的标准 Action：Intent. ACTION_VIEW，它只表示一个抽象的查看操作，但具体查看什么、启动哪个 Activity 来查看，Intent. ACTION_VIEW 并不知道——这取决于 Activity 的<intent-filter…>配置，只要某个 Activity 的<intent-filter…/>配置中包含该 ACTION_VIEW，该 Activity 就有可能被启动。

3. Data、Type 属性

Data 指数据，与动作和类别一样，这个配置可以出现多次或一次都不出现。例如：

```
<data android:scheme = "file"/>
<data android:scheme = "content"/>
<data android:mimeType = "Image/png"/>
```

每个数据<data>元素可以指定一个 URI 和一个数据类型(MIME 媒体类型)。URI

由 Scheme、Authority 和 Path 组成。

例如，下面的 URI：

```
Content://com.android.provider.MyProvider/user
```

其中，Scheme 是 Content，Authority 为 com. android. provider. MyProvider，路径为 /user。

当一个 Intent 对象中的 URI 被用来和一个 IntentFiler 中的 URI 比较时，实际上比较的是上面提到的 URI 的各个部分。例如，如果 IntentFiler 仅指定了 Scheme，所有 Scheme 的 URI 和这个 IntentFilter 都匹配；如果 IntentFilter 指定了 Scheme、Authority 但没有指定 Path，所有相同 Scheme 和 Authority 的 URI 可以匹配上，而不管它们的 Path；如果 IntentFilter 指定了 Scheme、Authority 和 Path，只有相同 Scheme、Authority 和 Path 的 URI 可以匹配上。当然，一个 IntentFilter 中的 Path 可以包含通配符，这样只需要部分匹配即可。

数据<data>元素的类型属性指定了数据的 MIME 类型，这在 IntentFilter 里比在 URI 题名为常见。Intent 对象和 IntentFilter 都可以使用一个 * 通配符指定子类型字段，例如，text/ * 或者 audio/ * 表示任何匹配的子类型。

数据<data>同时比较 Intent 对象和 IntentFilter 中指定的 URI 和数据类型，匹配规则如下：

- 一个既不包含 URI 也不包含数据类型的 Intent 对象，仅在 IntentFilter 中也没有指定任何 URI 和数据类型的情况下才能通过。
- 一个包含 URI 但没有数据类型的 Intent 对象，仅在它的 URI 和一个同样没有指定数据类型的 IntentFilter 中的 URI 匹配时才能通过。这通常发生在类似于 mailto 和 tel 这样的 URI 上，它们并不是引用实际数据。
- 一个包含数据类型但不包含 URI 的 Intent 对象，仅在这个 IntentFilter 中列举了同样的数据类型而且没有指定一个 URI 的情况下才能通过。
- 一个同时包含 URI 和数据类型(或者可从 URI 推断出数据类型)的 Intent 对象，如果它的类型和 IntentFilter 列举的类型相匹配，则可以通过；如果它的 URI 和这个 IntentFilter 中的一个 URI 相匹配或者它有一个内容(Content)或者文件(File) URI，而且这个 IntentFilter 仅指定了类型，那么它也能通过。

4. 额外(Extras)

当 Extras 是一组键值时，其中包含了应该传递给处理 Intent 的组件的额外信息。就像一些动作与特定种类的数据 URI 匹配，而一些与特定额外匹配。例如，动作为 ACTION_TIMEZONE_CHANGED 的 Intent 用 time-zone 额外来表示新时区；动作为 ACTION_HEADSET_PLUG 的 Intent 用 state 额外来表示耳机是否被插入，以及用 name 额外来表示耳机的类型。如果开发人员自定义一个 SHOW_COLOR 动作，则应该包含额外来表示颜色值。

Intent 对象中包含了多个 putXXX()方法(例如 putExtra()方法)用来插入不同类型的额外数据，也包含了多个 getXXX()方法(例如 getDoubleExtra()方法)来读取数据。这些方法与 Bundle 对象有些类似，实际上，额外可以通过 putExtras()和 getExtras()方法作为

Bundle 设置和读取。

5. 标记(Flags)

Flags 表示不同来源的标记,多用于指示 Android 系统怎样启动 Activity(例如 Activity 属于哪个 Task)以及启动后怎样对待(例如它是否属于近期的 Activity 列表)。所有标记都定义在 Intent 类中。

说明:所有标记都是整数类型。

6. Intent 对象的应用实例

下面通过几个实例来说明 Intent 对象的使用。

【例 2-3】 在 Activity 中使用包含项定义动作的隐式 Intent 启动另外一个 Activity。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 Intent1。

(2) 在 res\layout 文件夹中创建布局文件 main1. xml。具体操作为:

右击 res\layout 文件夹,在弹出的快捷菜单中选择"新建→文件"命令,弹出如图 2-12 所示的"新建文件"对话框,在"文件名"右侧的文本框中输入所创建的文件名,然后单击"完成"按钮,即可创建对应的布局文件。



图 2-12 "新建文件"对话框

打开 res\layout 目录下的 main. xml 布局文件,在布局文件 main1. xml 中保留一个按钮,并修改其默认属性。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp"
    android:orientation = "vertical" >
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/button"
        android:textColor = "@android:color/black"/>
</LinearLayout >
```

(3) 在布局文件中添加文本框控件来显示字符串，并修改默认属性。所创建的布局文件名为 main2.xml，修改后的布局代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp"
    android:orientation = "vertical" >
    < TextView
        android:id = "@ + id/textView"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/text"
        android:textColor = "@android:color/black"
        android:textSize = "25px" />
</LinearLayout >
```

注意：在以上代码中设置了图形的背景图为 kp，该图放置在 res\drawble-hdpi 中。

(4) 编写 Main1Activity 类，获得布局文件中的按钮控件并为其添加单击事件监听器。在监听器中传递包含的隐式 Intent：

```
package fs.intent1;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class Main1Activity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main1);                    // 设置页面布局
        Button button = (Button) findViewById(R.id.button1);        // 通过 id 值获得按钮对象
        button.setOnClickListener(new View.OnClickListener()
{// 为按钮添加单击事件监听器
                public void onClick(View v) {
                    Intent intent  =  new Intent();     // 创建 Intent 对象
                    intent.setAction(Intent.ACTION_VIEW);        // 为 Intent 设置动作
```

```
                startActivity(intent);          // 将 Intent 传递给 Activity
            }
        });
    }
}
```

注意：在 Main1Activity 类的代码中并没有指定将 Intent 对象传递给哪个 Activity。

(5) 编写 Main2Activity 类，具体操作为：

首先选择 src\fs.intent1 文件夹右击，在弹出的快捷菜单中选择"新建→类"命令，弹出"新建 Java 类"对话框，如图 2-13 所示。然后在"名称"右侧的文本框中输入对应的类名，单击"超类"右侧的"浏览"按钮，弹出"选择超类"对话框，如图 2-14 所示。在"选择类型"右侧的文本框中输入 View，即弹出对应的选择，在此选择"View-android. view "定义 View 类，至此完成新类的创建。



图 2-13 "新建 Java 类"对话框

自定义 Main2Activity 类，仅为其设置布局文件：

```
package fs.intent1;
import android.app.Activity;
import android.os.Bundle;
public class Main2Activity extends Activity{
    @Override
```

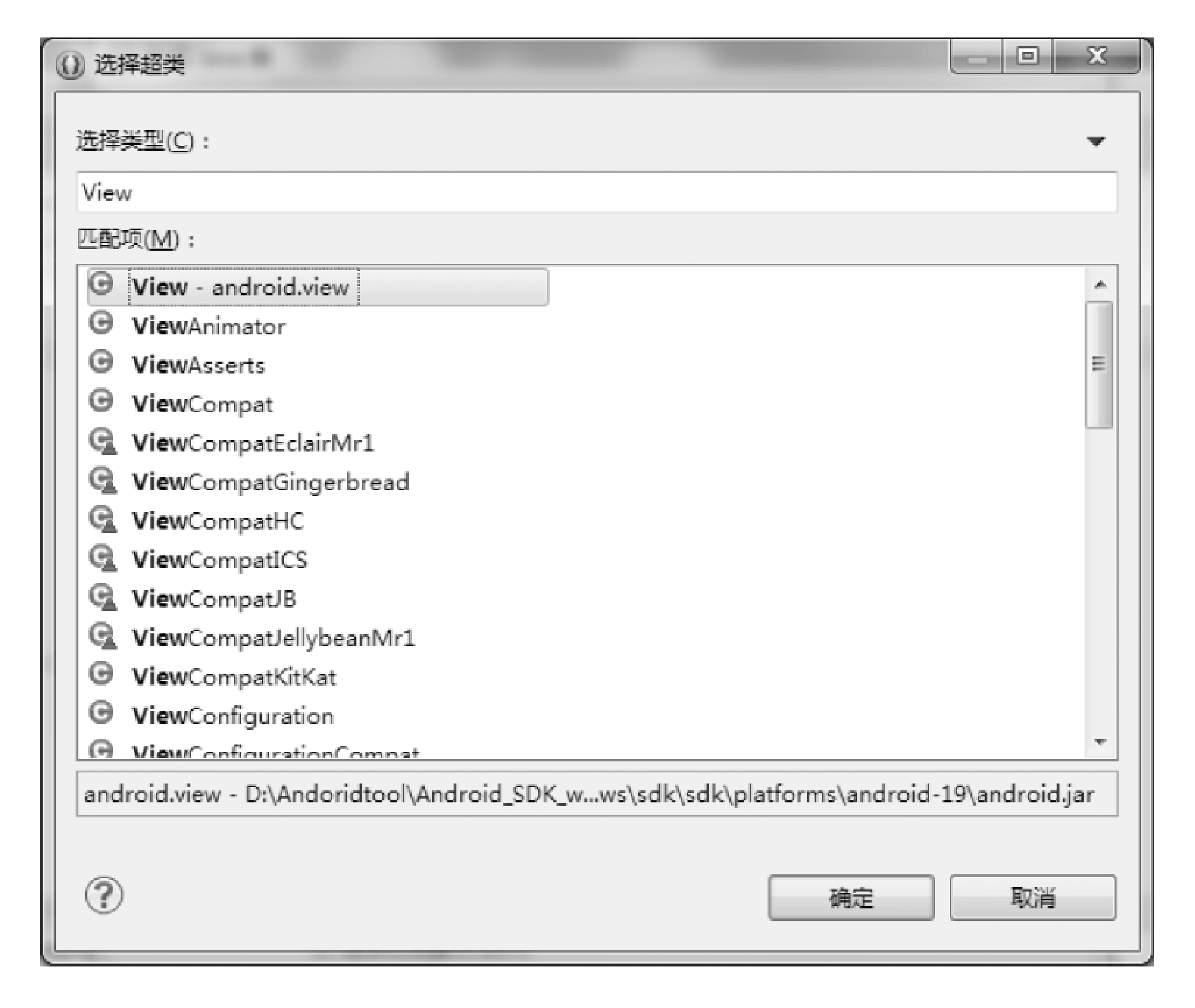


图 2-14 “选择超类”对话框

```
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);              // 设置页面布局
    }
}
```

(6) 双击 AndroidManifest.xml 文件，为两个 Activity 设置不同的 Intent 过滤器。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.intent1"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < Activity
            android:name = "fs.intent1.Main1Activity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </Activity >
```

```
        <Activity
    android:name = "fs.intent1.Main2Activity"
    android:label = "@string/app_name" >
    <intent-filter>
        <action android:name = "android.intent.action.VIEW" />
        <category android:name = "android.intent.category.DEFAULT" />
    </intent-filter>
  </Activity>
 </application>
</manifest>
```

(7) 设置标题。打开 res\values\string.xml 文件，该文件用于设置界面标题，其代码为：

```
<?xml version = "1.0" encoding = "utf-8"?>
<resources>
    <string name = "app_name">Intent1</string>
    <string name = "button">转到下一个 Activity</string>
    <string name = "text">第二个 Activity</string>
</resources>
```

(8) 运行程序，在打开的界面中单击“转到下一个 Activity”按钮，显示如图 2-15 所示的界面。

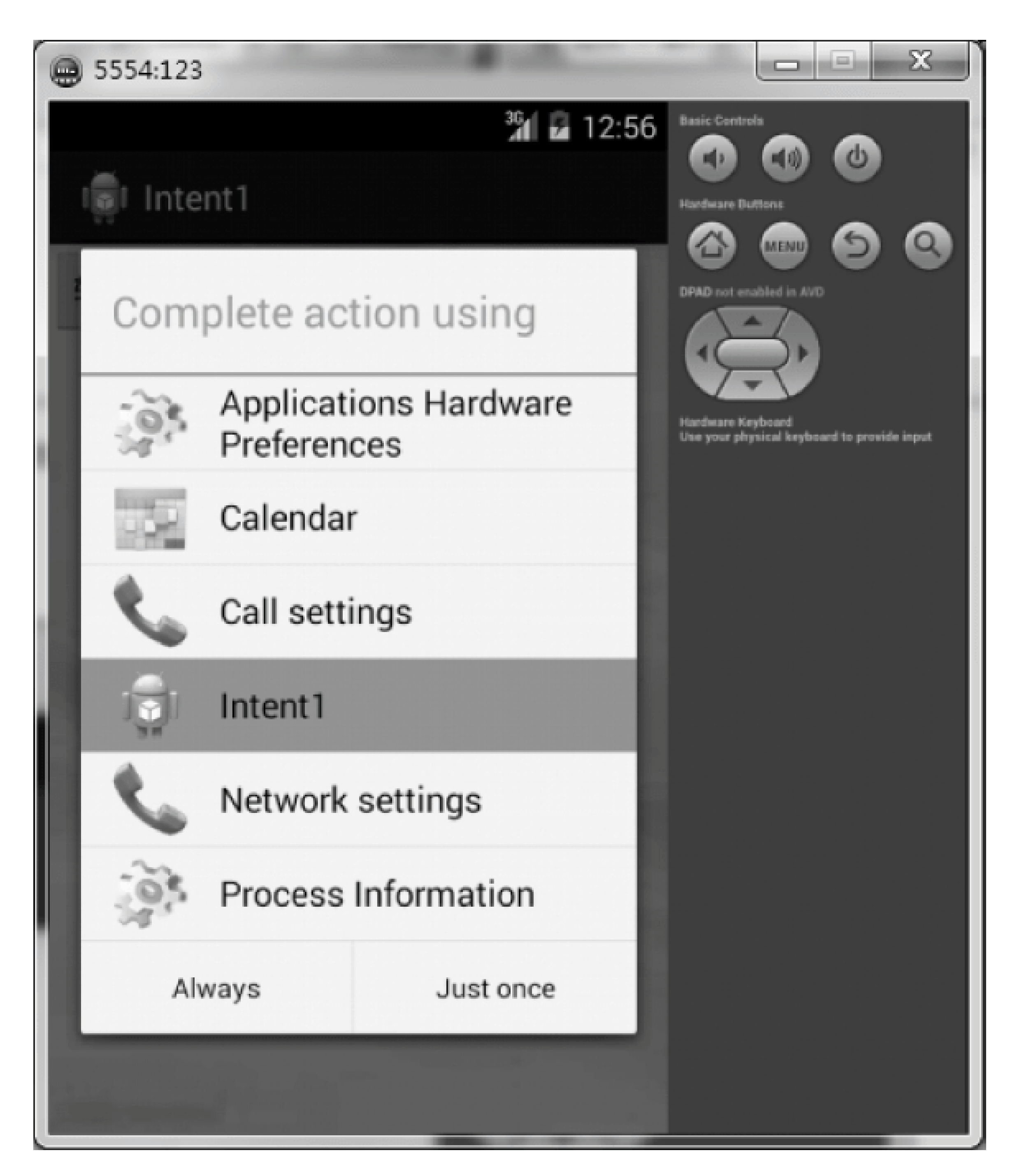


图 2-15　选择发送方式界面

选择图 2-15 中的 Intent1 选项，即跳转到第二个 Activity，界面如图 2-16 所示。

说明：由于有多种匹配 ACTION_VIEW 的方式，因此需要用户进行选择。

在例 2-3 中介绍了使用系统中预定义的动作来定义 Intent，开发人员还可以根据需要自定义动作。本实例将在例 2-3 的基础上进行修改，使用自定义动作来启动隐式 Intent。

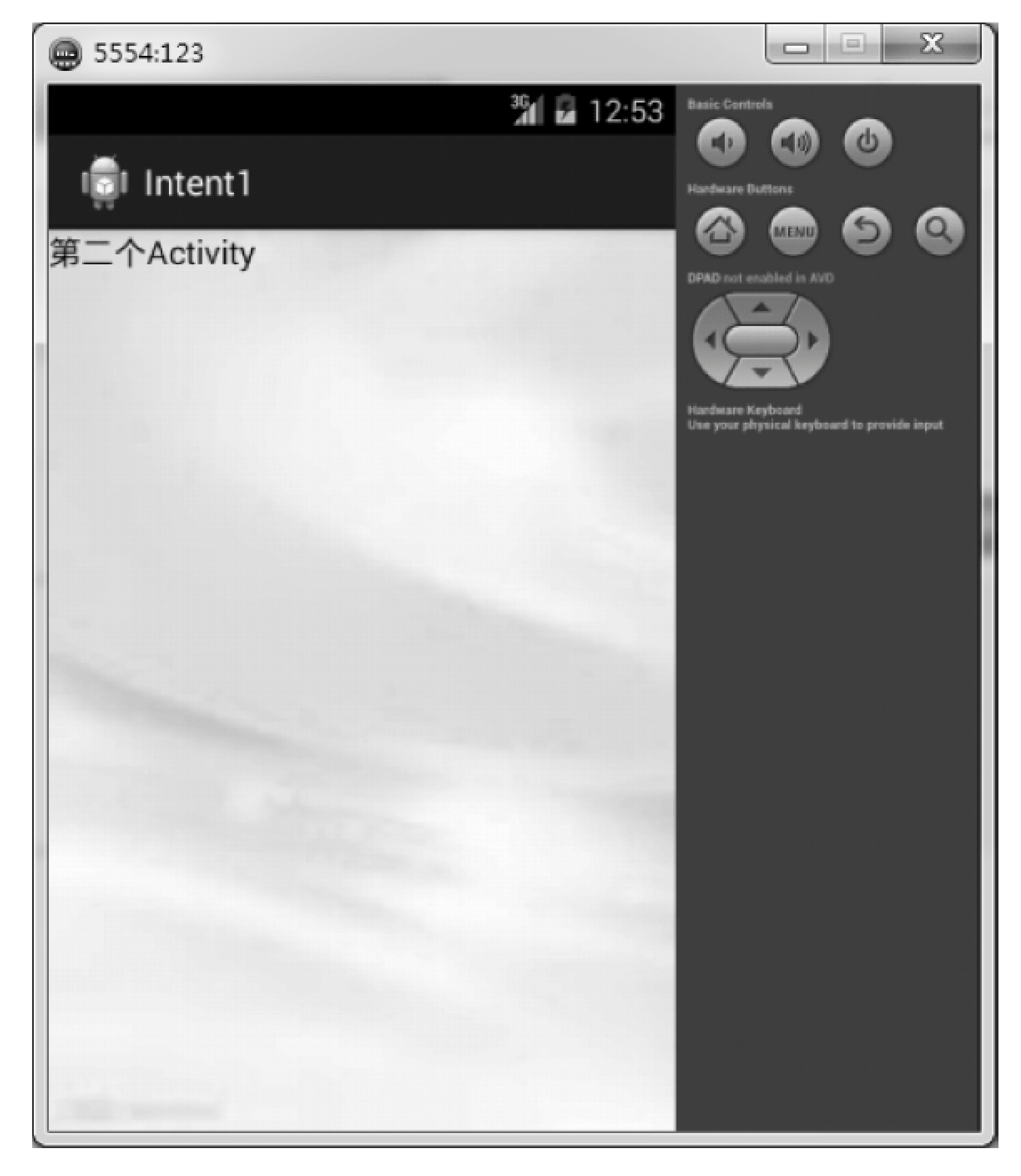


图 2-16　第二个 Activity 界面

【例 2-4】 在 Activity 中使用包含自定义动作的隐式 Intent 启动另外一个 Activity。其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 Intent2。

(2) 在例 2-3 的基础上将 Main1Activity 类的代码修改为:

```
public class Main1Activity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main1);                    //设置页面布局
        Button button = (Button) findViewById(R.id.button1);   //通过 id 值获得按钮对象
            button.setOnClickListener(new View.OnClickListener()
            {//为按钮添加单击事件监听器
                public void onClick(View v) {
                    Intent intent  =  new Intent();      //创建 Intent 对象
                    intent.setAction("text_action"); //为 Intent 设置动作
                    startActivity(intent);           //将 Intent 传递给 Activity
                }
            });
      }
}
```

(3) 将 AndroidMainfest.xml 文件代码修改为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.intent1"
```

```
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < Activity
            android:name = "fs.intent1.Main1Activity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </Activity >
                < Activity
            android:name = "fs.intent1.Main2Activity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "test_action" />
                < category android:name = "android.intent.category.DEFAULT" />
            </intent - filter >
        </Activity >
    </application >
</manifest >
```

运行程序，效果如图 2-17 所示。单击“转到下一个 Activity”按钮，效果如图 2-18 所示。此时并没有让用户选择处理隐式 Intent 的组件，而是直接跳转到第二个 Activity。

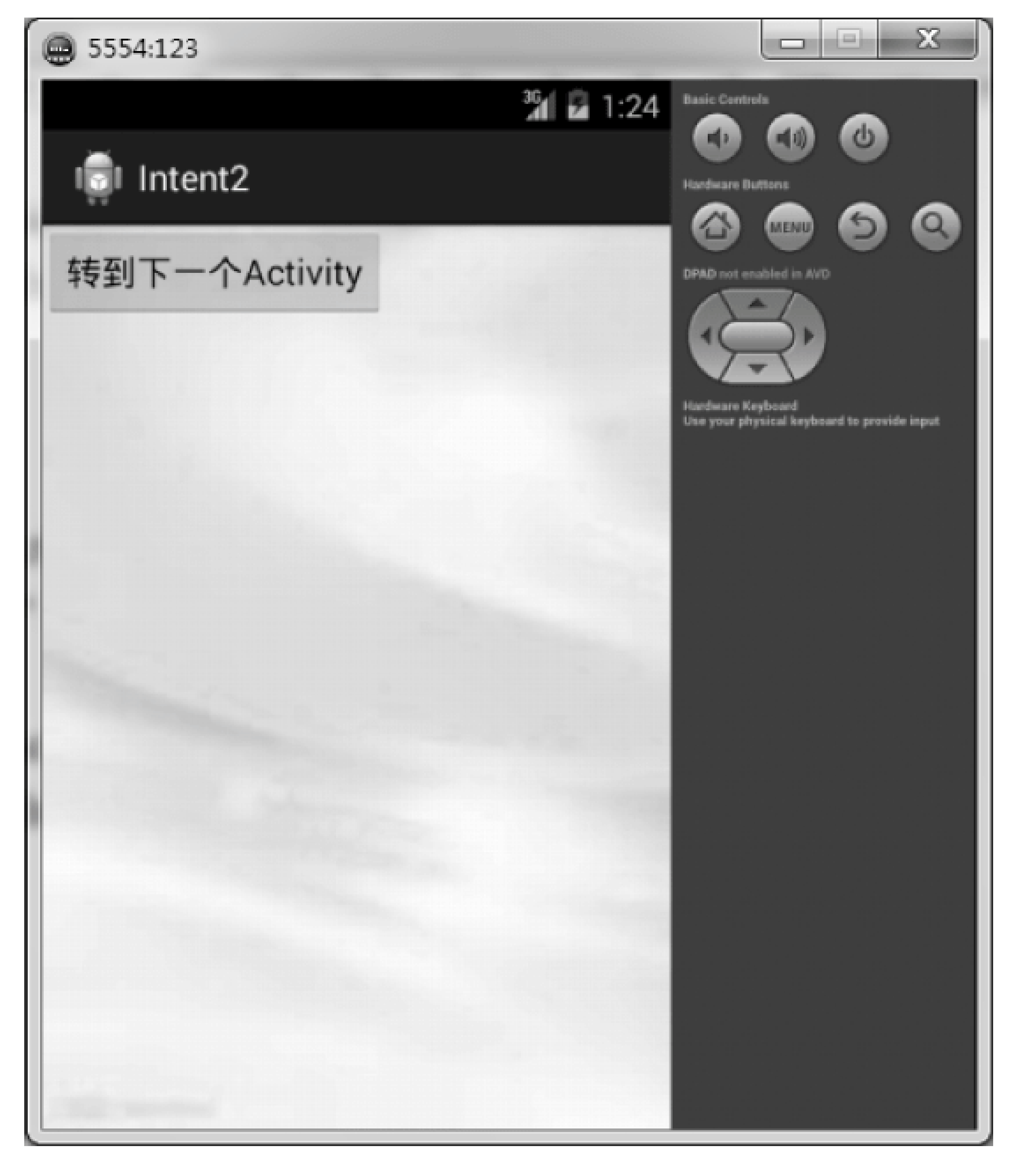


图 2-17　第一个 Activity 界面

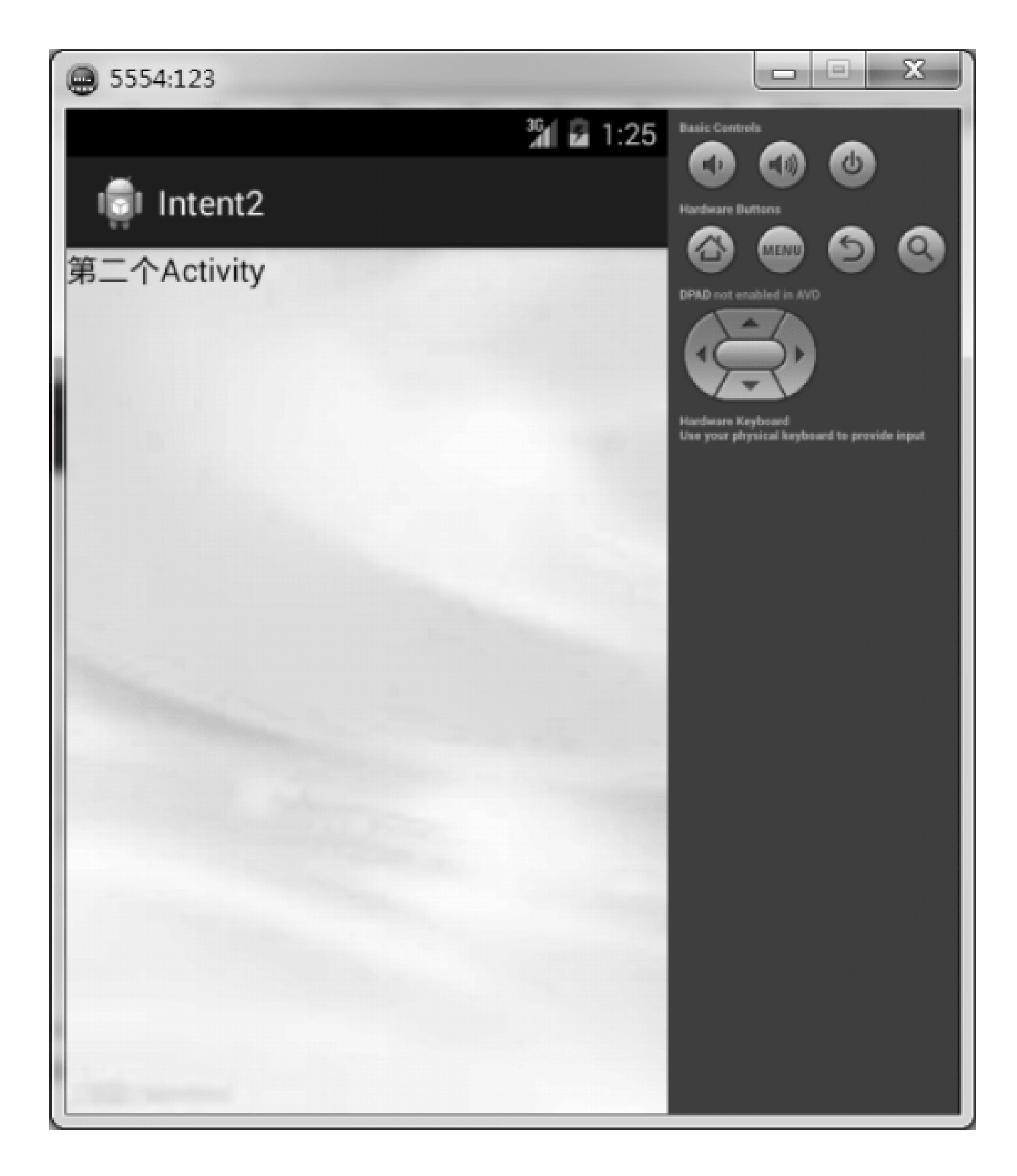


图 2-18　第二个 Activity 界面

2.4.2　Intent 经典实例

在前面已对 Intent 的概述和应用做了介绍，下面通过一个经典实例来介绍 Intent 的实际应用。

【例 2-5】　使用 Intent 拨打电话。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 Intent3，实现拨打电话功能。

(2) 在 res\layout 文件夹中打开布局文件 main.xml，在文件中添加一个文本框和一个按钮，并修改其默认属性。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp"
    android:orientation = "vertical" >
    < EditText
        android:id = "@ + id/editText1"
        android:layout_width = "276dp"
        android:layout_height = "wrap_content"
        android:layout_alignParentTop = "true"
        android:layout_centerHorizontal = "true"
        android:layout_gravity = "center_vertical"
        android:layout_marginTop = "42dp"
        android:ems = "10"
        android:inputType = "phone" />
    < Button
```

```
        android:id = "@ + id/button1"
        android:layout_width = "123dp"
        android:layout_height = "79dp"
        android:layout_gravity = "center_horizontal"
        android:layout_marginTop = "85dp"
        android:text = "拨打"
        android:textColor = "@android:color/black" />
</LinearLayout >
```

(3) 在 src\ E_. intent3 文件中打开 MainActivity 文件并进行编写,它从页面中获得用户输入的电话号码,通过为按钮添加单击事件监听器来完成拨号功能:

```
package E_. intent3;
import android. app. Activity;
import android. content. Intent;
import android. net. Uri;
import android. os. Bundle;
import android. view. View;
import android. widget. Button;
import android. widget. EditText;
import android. app. Activity;
import android. view. Menu;
public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);                //设置页面布局
        EditText numberTV = (EditText) findViewById(R. id. editText1);
         //通过 id 值获得文本框对象
        final String number = numberTV. getText(). toString();      //获得输入的电话号码
        Button dial = (Button) findViewById(R. id. button1);         //通过 id 值获得按钮对象
        dial. setOnClickListener(new View. OnClickListener()
        {
            public void onClick(View v)
            {
                Intent intent = new Intent();          //创建 Intent 对象
                intent. setAction(Intent. ACTION_DIAL);//设置 Intent 动作
                intent. setData(Uri. parse("tel:" + number));       //设置 Intent 数据
                startActivity(intent);                 //将 Intent 传递给 Activity
                }
            });
        }
}
```

(4) 修改 AndroidMainfest. xml 文件,增加拨打电话的权限:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas. android. com/apk/res/android"
    package = "E_. intent3"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
```

```
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < Activity
            android:name = ".MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </Activity >
    </application >
< uses - permission android:name = "android.permission.CALL_PHONE"/>
</manifest >
```

其他文件的代码修改可参照例 2-3 进行。

运行程序,效果如图 2-19 所示。在文本框中输入需要拨打的电话,单击“拨打”按钮即可完成拨号功能。



图 2-19　拨打电话界面

【例 2-6】　使用 Intent 打开网页。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 Intent4。

(2) 在 res\layout 文件夹中打开布局文件 main.xml,在布局文件中添加一个按钮,并修改默认属性:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp"
```

```
    android:orientation = "vertical" >
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "170dp"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center"
        android:text = "打开网页" />
</LinearLayout >
```

(3) 编写 MainActivity 文件,通过为按钮添加事件监听器来完成打开网页功能:

```
public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                //设置页面布局
        Button button = (Button) findViewById(R.id.button1);      //通过 id 值获得按钮对象
        button.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v)
            {
                Intent intent = new Intent();         //创建 Intent 对象
                intent.setAction(Intent.ACTION_VIEW);//设置 Intent 动作
                    //设置 Intent 数据
intent.setData(Uri.parse("http://www.hao123.com/?tn = 95235286_hao_pg"));
                    startActivity(intent);            //将 Intent 传递给 Activity
            }
        });
    }
}
```

其他文件采用默认值,然后运行程序,效果如图 2-20(a)所示。单击图中的“打开网页”按钮,效果如图 2-20(b)所示。



(a) 打开网页默认界面



(b) hao123主页

图 2-20 打开网页实例效果

2.5 Adapter 对象

Android Adapter 是将数据绑定到 UI 界面上的桥接类。Adapter 负责创建和显示每个项目的子 View 和提供对下层数据的访问。支持 Adapter 绑定的 UI 控件必须扩展 Adapter View 抽象类。创建自己的继承自 AdapterView 的控件和创建新的 Adapter 类来绑定它们是可能的。

Android 系统本身提供了两种现成的 Adapter 用户。

(1) ArrayAdapter：ArrayAdapter 是一个绑定 View 到一组对象的通用类。默认情况下，ArrayAdapter 绑定每个对象的 toString 值到 Layout 中预先定义的 TextView 空间上。构造函数允许用户使用更加复杂的 Layout 或者通过重写 getView 方法来扩展类，从而使用 TextView 的替代物。

(2) SimpleCursorAdapter：SimpleCursorAdapter 绑定 View 到 Content Provider 查询返回的游标上。指定一个 XML Layout 定义，然后将数据集的每一列的值绑定到 Layout 中的一个 View。

2.5.1 Adapter 绑定

在 Android 中可以使用 Adapter 对数据进行绑定。将 Adapter 应用到继承自 AdapterView 类上，需要调用 View 的 setAdapter 方法传入一个 Adapter 实例。

以下代码为自定义的 Adapter 类，可实现更多复杂的 UI 界面和数据绑定：

```
public class MyAdapter extends SimpleAdapter
{
     private LayoutInflater mInflater;
     private Context context;
     private List<Map<String,Object>> list;
     private int resource;
     private String[] tags;
     private int[] ids;
     public MyAdapter(Context context, List<Map<String,Object>> items, int resource,String[]
tags,int[] ids)
    {
          super(context, items, resource, tags, ids);
          this.mInflater = LayoutInflater.from(context);
          this.context = context;
          this.list = items;
          this.resource = resource;
          this.tags = tags;
          this.ids = ids;
     }
    public int getCount()
    {
          return list.size();
     }
     public Object getItem(int position)
    {
          return list.get(position);
```

```
    }
    public long getItemId(int position)
    {
        return position;
    }
    public View getView(final int position, View convertView, ViewGroup parent)
    {
        convertView = super.getView(position, convertView, parent);
        if (convertView == null)
        {
            Toast.makeText(context, "this is null", 2000).show();
        } else {}
        ImageView more = (ImageView) convertView.findViewById(R.id.iv_more);
        more.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View arg0)
            {
                Intent intent = new Intent(context, VehicleInfoActivity.class);
                intent.putExtra("vehicleID", VehicleListActivity.idList.get(position));
                intent.putExtra("CameraID", "0");
                Toast.makeText(context, "sssssss", 2000).show();
                context.startActivity(intent);
            }
        });
        return convertView;
    }
}
```

以上程序主要重载 getCount 方法、getView 方法、getItem 方法、getItemId 方法。其中，参数 context 为传入的上下文 Activity，参数 items 为绑定数据的列表，参数 resource 为 Layout 布局 ID，参数 tags 为绑定数据的 key，ids 为 Item 中对应 key 的资源 id。在 getView 中自定义 Item 里面的事件监听，自定义新的重载后的显示界面返回 convertView。如果需要使用父类 SimpleAdapter 显示效果，需要调用 super.getView 方法为 convertView 赋值。

2.5.2 ArrayAdapter 与 SimpleCursorAdapter

ArrayAdapter 与 SimpleCursorAdapter 类似，SimpleCursorAdapter 也是集成 Adapter。ArrayAdapter 负责把一个字符串数组中的数据填充到一个 ListView 当中，而对应的 SimpleCursorAdapter 负责把 Cursor 里面的内容填充到 ListView 当中。通过 SimpleCursorAdapter 可以把数据库中一列的数据和 ListView 中的一排对应起来。

1. ArrayAdapter 的使用

这里通过一个实例的实现过程来讲解 CursorAdapter 的使用方法，其实现过程如下：

(1) 在 Eclipse 环境下建立一个名为 A_Adapter 的工程。

(2) 编写布局文件，布局一个列表视图。打开 res\Layout 目录下的 main.xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
```

```
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:background = "@drawable/bj">
    <ListView
        android:id = "@ + id/list"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content" />
    </LinearLayout>
```

(3) 编写 Activity 文件。打开 src\com.example.a_adapter 包下的 MainActivity.java 文件,代码为:

```
package com.example.a_adapter;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
public class MainActivity extends Activity
{
    ListView lv;
    String[] value = {
            "JAN","FEB","MAR","APR",
            "MAY","JUN","JUL","AUG","SEP","OCT","NOV","DEC "
    };
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        lv = (ListView) findViewById(R.id.list);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,android.R.layout.
simple_list_item_1,value);
        lv.setAdapter(adapter);
    }
}
```

在以上代码中,先定义了一个 String[],里面保存了 12 个月份的英文缩写,然后通过 ListView 将数组中的数据显示出来。运行程序,效果如图 2-21 所示。



图 2-21　ArrayAdapter 实现效果

2. SimpleCursorAdapter 的使用

SimpleCursorAdapter 允许用户绑定一个游标的列到 ListView 上,并使用自定义的 Layout(布局)显示每个项目。要实现 SimpleCursorAdapter 的创建,需要传入当前的上下文、一个 Layout 资源、一个游标和两个数组。其中一个数组包含使用的列的名字,另一个数组包含 View 中的资源 id,用于显示相应列的数据值。

这里通过一个实例的实现过程来演示 SimpleCursorAdapter 的用法,其实现过程如下:

(1) 在 Eclipse 环境下建立一个名为 S_Adapter 的

工程。

(2) 编写主布局文件,布局一个列表视图。打开 res\Layout 目录下的 main. xml 文件,代码为:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bj">
    <ListView android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:id="@+id/ListView01" />
</LinearLayout>
```

(3) 编写用来确定 ListView 中的每行怎么显示的布局文件。在 res\Layout 目录下新建一个名为 listview_row. xml 的文件:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RelativeLayout01"
    android:layout_width="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="wrap_content"
    android:paddingTop="4dip"
    android:paddingBottom="4dip"
    android:paddingLeft="12dip"
    android:paddingRight="12dip">
<ImageView
    android:paddingTop="12dip"
    android:layout_alignParentRight="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/itemImage"/>
<TextView
    android:layout_height="wrap_content"
    android:textSize="20dip"
    android:layout_width="fill_parent"
    android:id="@+id/itemTitle"/>
<TextView
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:layout_below="@+id/itemTitle"
    android:id="@+id/itemText" />
</RelativeLayout>
```

(4) 编写 Activity 文件。打开 src\com. example. s_adapter 包下的 MainActivity. java 文件,代码为:

```
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //绑定 Layout 里面的 ListView
        ListView list = (ListView) findViewById(R.id.ListView01);
        //生成动态数组,加入数据
        ArrayList < HashMap < String, Object >> listItem = new ArrayList < HashMap < String,
Object >>();
        int[] images = new int[]{android.R.drawable.ic_menu_add,android.R.drawable.ic_menu_
delete,android.R.drawable.ic_menu_edit,android.R.drawable.ic_menu_view};
        for(int i = 0;i < 4;i++)
        {
            HashMap < String, Object > map = new HashMap < String, Object >();
            map.put("itemImage", images[i]);          //图像资源的 id
            map.put("itemTitle", "Title " + i);
            map.put("itemText", "this is Text " + i);
            listItem.add(map);
        }
        //生成适配器的 Item 和动态数组对应的元素
        SimpleAdapter listItemAdapter = new SimpleAdapter(this,listItem,//数据源
            R.layout.listview_row,                    //ListItem 的 XML 实现
            //动态数组与 ImageItem 对应的子项
            new String[] {"itemImage","itemTitle", "itemText"},
            //ImageItem 的 XML 文件里面的一个 ImageView、两个 TextView id
            new int[] {R.id.itemImage,R.id.itemTitle,R.id.itemText}
        );
        //添加并且显示
        list.setAdapter(listItemAdapter);
        //添加单击
        list.setOnItemClickListener(new OnItemClickListener()
        {
          public void onItemClick(AdapterView <?> arg0, View arg1, int arg2,long arg3)
            {
                setTitle("单击第" + arg2 + "个项目");
            }
        });
      //添加长按单击
        list.setOnCreateContextMenuListener(new OnCreateContextMenuListener()
        {
            public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)
            {
                menu.setHeaderTitle("长按菜单 - ContextMenu");
                menu.add(0, 0, 0, "弹出长按菜单 0");
                menu.add(0, 1, 0, "弹出长按菜单 1");
            }
        });
    }
    //长按菜单响应函数
    @Override
    public boolean onContextItemSelected(MenuItem item)
    {
        setTitle("单击了长按菜单里面的第" + item.getItemId() + "个项目");
        return super.onContextItemSelected(item);
    }
}
```

运行程序,初始效果如图 2-22 所示。当单击图 2-22 中的选项时,效果如图 2-23 所示。




图 2-22　初始效果



图 2-23　选择选项

2.6　消息传递机制

Android 的消息传递机制是另一种形式的"事件处理",这种机制主要是为了解决 Android 应用的多线程问题——Android 平台不允许 Activity 新启动的线程访问该 Activity 里的界面组件,这样就会导致新启动的线程无法动态改变界面组件的属性值。但在实际 Android 应用开发中,尤其是涉及动画的游戏开发中,需要让新启动的线程周期性地改变界面组件的属性值,这就需要借助于 Handler 的消息传递机制来实现了。

每一个应用程序(APK)都是一个单独的进程,运行于单独的 Dalvik 虚拟机实例中,再运行于单独的 Linux 进程中。每一个进程默认只有一个线程,即 UI 主线程,因为它是以 UI 界面更新为主要任务的主线程。

同样继承于 Context 的 Activity 和 Service 都是跑在同一个线程里的,即 UI 主线程。这样它们之间是相互阻塞的,当 Service 运行较为费时的工作,而 Activity 上的 UI 界面需要更新导致程序卡住时,就会导致程序被系统终止。解决办法是 Service 需要开辟更多的线程来运行费时的工作,例如播放音乐、网络下载等。

那么开辟的线程怎样与 UI 主线程互相协调工作呢?这就需要用到 Handler 了。最主要的作用是管理线程间的消息通信。

UI 主线程初始化时自带一个消息队列(MessageQueen),需要用 Looper 进行管理。Looper 的主要作用是循环迭代 MessageQueen,加入新的 Message,当轮到这条消息时再发送出去,而 Handler 就可以直接操作 Looper 了。例如:

```
Handler mHandler = new Handler(Looper.getMainLooper);
```

就是用 mHandler 对象控制 UI 主线程的 Looper 对象,也就是控制了 MessageQueen。

```
Handler mHandler = new Handler(Looper.myLooper);
```

就是控制当前线程(也可能是 UI 主线程)的 MessageQueen。

那么,Handler 控制了各个线程的 MessageQueen 就可以实现主线程与子线程的相互通信、传递消息队列、分派任务等复杂的工作了。

Handler 还有一个用处,就是缓和任务冲突。当 UI 急需更新但是任务却处在繁忙状态时,为了避免被系统杀死,这时最好利用 Handler 发送简单消息通知 UI 主线程更新界面,UI 主线程在运行时会在空闲时取出消息解析运行,这样就不会造成任务冲突了。但是这样可能会造成运行不流畅,最好的办法还是开辟一个新的线程来分摊工作。

有些没有界面的任务(比如 Service 运行时)需要更新 UI 界面,例如弹出对话框(Dialog),那就要通过 Handler 来发送消息,通知 UI 线程更新界面,这样才能弹出对话框。

Handler 类包含以下方法用于发送、处理消息。

- void handleMessage(Message msg):处理消息的方法。该方法通常用于重写。
- final boolean hasMessages(int what):检查消息队列中是否包含 what 属性为指定值的消息。
- final boolean hasMessages(int what,Object object):检查消息队列中是否包含 what 属性为指定值且 object 属性为指定对象的消息。
- Message obtainMessage():获取消息。
- sendEmptyMessage(int what):发送空消息。
- final boolean sendEmptyMessageDelayed(int what,long delayMillis):指定多少毫秒之后发送空消息。
- final boolean sendMessage(Message msg):立即发送消息。
- final boolean sendMessageDelayed(Message msg,long delayMillis):指定多少毫秒之后发送的消息。

借助上面这些方法,程序可以方便地利用 Handler 进行消息传递。

下面通过一个实例来说明 Handler 类的使用。

【例 2-7】 利用 Android 创建多彩的闪烁灯。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 colorright。

(2) 打开 res\layout 目录下的布局文件 main.xml,将默认添加的 TextView 控件删除,并为默认添加的线性布局管理器设置 ID 属性:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/bj1"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
</LinearLayout >
```

(3) 在 res\values 目录下创建一个颜色资源文件,命名为 color.xml,并在该文件中定义 7 个颜色资源,名称依次为 color1、color2、…、color7,颜色值分别为赤、橙、黄、绿、青、蓝、紫。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < color name = "color1"># ffff0000 </color >
    < color name = "color2"># ffff6600 </color >
    < color name = "color3"># ffffff00 </color >
    < color name = "color4"># ff00ff00 </color >
    < color name = "color5"># ff00ffff </color >
    < color name = "color6"># ff0000ff </color >
    < color name = "color7"># ff6600ff </color >
</resources >
```

(4) 打开默认创建的 MainActivity,其中对相应的颜色已做说明。文件的主要代码为:

```
public class MainActivity extends Activity
{   //声明程序中所需的成员变量
    private Handler handler;                        //创建 Handler 对象
    private static LinearLayout linearLayout;       //整体布局
    public static TextView[] tv  =  new TextView[14];//TextView 数组
    int[] bgColor = new int[]{R.color.color1,R.color.color2,R.color.color3,
    R.color.color4,R.color.color5,R.color.color6,R.color.color7};     //使用颜色资源
    private int index = 0;                          //当前的颜色值
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //首先获取线性布局管理器,然后获取屏幕的高度,接着通过一个 for 循环创建 14 个文本
        //框控件,并添加到线性布局管理器中
        linearLayout = (LinearLayout)findViewById(R.id.bj1);         //获取线性布局管理器
        int height = this.getResources().getDisplayMetrics().heightPixels;
        //获取屏幕的高度
        for(int i = 0;i < tv.length;i++){
            tv[i] = new TextView(this);                 //创建一个文本框对象
         //设置文本框的宽度
            tv[i].setWidth(this.getResources().getDisplayMetrics().widthPixels);
    tv[i].setHeight(height/tv.length);              //设置文本框的高度
            linearLayout.addView(tv[i]);                //将 TextView 控件添加到线性布局管理器中
        }
        //创建一个 Handler 对象, 在重写的 handleMessage()方法中为每个文本框设置背景颜色,
        //该背景颜色从颜色数组中随机获取
        handler  =  new Handler() {
            @Override
            public void handleMessage(Message msg) {
                int temp = 0;                           //临时变量
                if (msg.what  ==  0x101) {
                    for(int i = 0;i < tv.length;i++){
                    //产生一个随机数
                        temp = new Random().nextInt(bgColor.length);
                                                        //去掉重复的并且相邻的颜色
                        if(index == temp){
                            temp++;
                            if(temp == bgColor.length){
                                temp = 0;
                            }
                        }
                        index = temp;
        tv[i].setBackgroundColor(getResources().getColor(bgColor[index]));
                                                                  //为文本框设置背景
```

```
                }
            }
            super.handleMessage(msg);
        }
    };
    //创建并开启一个新线程,在重写的 run()方法中实现一个循环。在该循环中,先获取一个
    //Message 对象,并为其设置一个消息标识,然后发送消息,最后让线程休眠 1 秒钟
    Thread t = new Thread(new Runnable()
    {
        @Override
        public void run() {
            while (!Thread.currentThread().isInterrupted()) {
                Message m = handler.obtainMessage();          //获取一个 Message
                m.what = 0x101;                    //设置消息标识
                handler.sendMessage(m);            //发送消息
                try {
                    Thread.sleep(new Random().nextInt(1000));      //休眠 1 秒钟
                } catch (InterruptedException e) {
                    e.printStackTrace();           //输出异常信息
                }
            }
        }
    });
    t.start();                                     //开启线程
  }
}
```

(5) 在 AndroidMainfest.xml 文件的<Activity>标记中设置 android:theme 属性,实现全屏显示:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.colorright"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <Activity
            android:name = ".MainActivity"
            android:label = "@string/app_name"
            android:theme = "@android:style/Theme.Black.NoTitleBar">
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </Activity>
    </application>
</manifest>
```

运行程序，将全屏显示一个多彩的闪烁灯，即它可以不断地变换颜色，效果如图 2-24 所示。

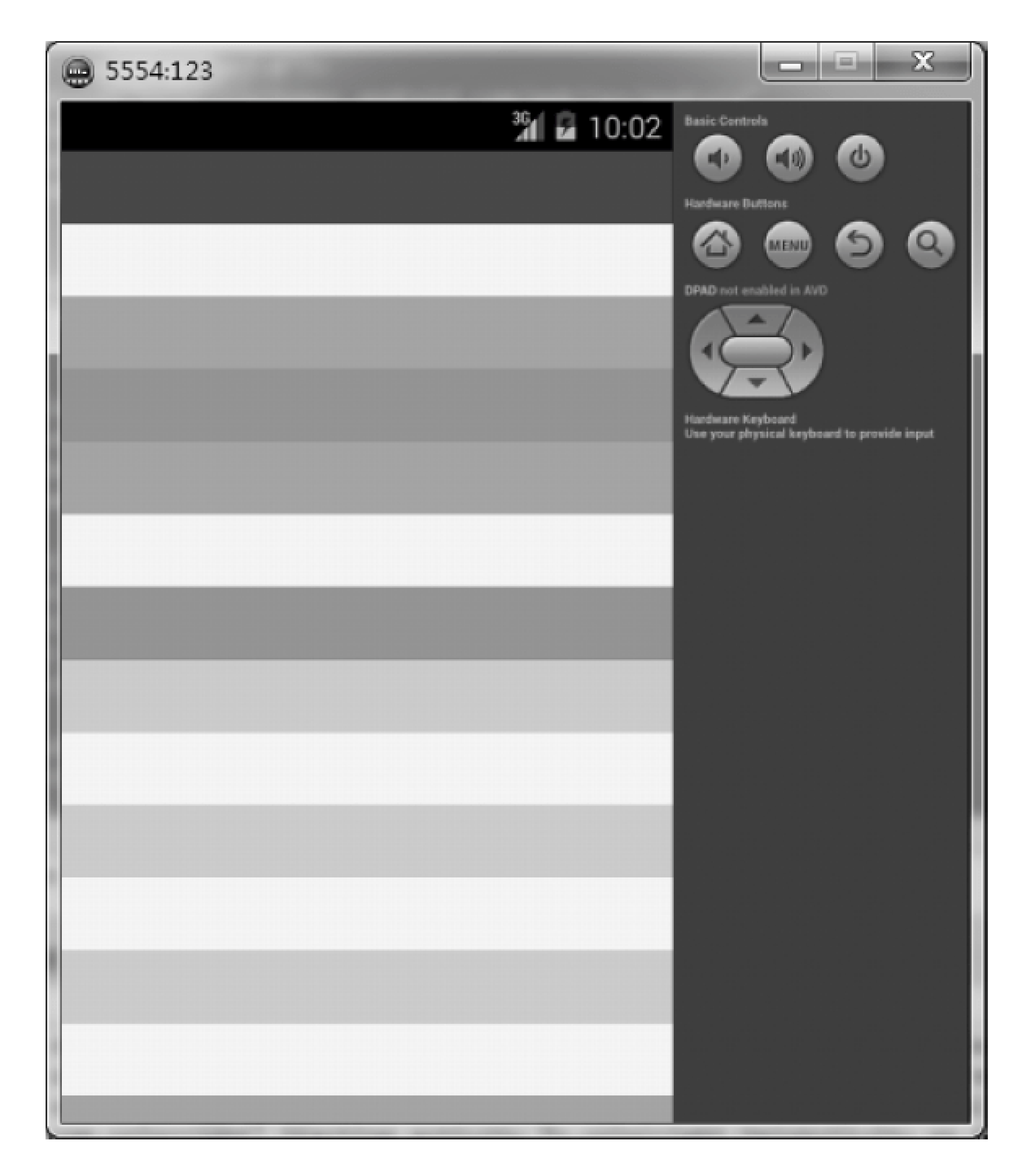


图 2-24　多彩闪烁灯

第3章 Android 布局

通过前面两章的学习，相信读者已经对 Android 有了一定的了解，本章将学习 Android 开发中的一项重要内容——Android 布局。

3.1 UI 界面

用户界面(User Interface，UI)设计是 Android 应用开发的一项重要内容。在进行布局介绍时，首先需要了解页面中的 UI 元素怎样呈现给用户，以及怎样控制 UI 界面。Android 提供了 4 种控制 UI 界面的方法。

3.1.1 布局文件控制 UI

Android 提供了一种非常简单、方便的方法用于控制 UI 界面。该方法采用 XML 文件进行界面布局，从而将界面布局的代码和逻辑控制的 Java 代码分离开来，使程序的结构更加清晰、明了。

使用 XML 布局文件控制 UI 界面可以分为两个关键步骤。

(1) 在 Android 应用文件的 res\layout 目录下编写 XML 布局文件，可以采用任何符合 Java 命名规则的文件名。创建后，R.java 会自动收录该布局资源。

(2) 在 Activity 中使用以下 Java 代码显示 XML 文件中布局的内容。

```
setContentView(R.layout.main);
```

其中，main 为 XML 布局文件的文件名。

下面通过一个简单实例演示怎样使用 XML 布局文件控制 UI 界面。

【例 3-1】 使用 XML 布局文件实现游戏的开始界面。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li3_1UI。

(2) 打开 res\layout 目录下的布局文件 main.xml。在该文件中采用帧布局(FrameLayout)，添加两个 TextView 控件，第 1 个用于显示提示文字，第 2 个用于在窗体的正中间位置显示“开始游戏”按钮。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj2"      >
    < TextView
        android:id = "@ + id/textView1"
```

```
        android:layout_width = "277dp"
        android:layout_height = "wrap_content"
        android:text = "@string/title" />
    < TextView
        android:id = "@ + id/textView2"
        android:layout_width = "153dp"
        android:layout_height = "38dp"
        android:layout_gravity = "center"
        android:text = "@string/start"   />
</FrameLayout >
```

在布局文件 main. xml 中，通过设置布局管理器的 android:background 属性可以为窗体设置背景图片，该图片放置在 res\drawble-hdpi 等几个文件中的其中一个。通过设置具体组件的 style 属性，可以为组件设置样式。使用 android:layout_gravity＝"center"，可以使该组件在帧布局中居中显示。

（3）选择 res\values 目录下的 strings. xml 文件，在该文件中添加一个用于定义开始按钮内容的常量，名称为 start，内容为“开始游戏”：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "title">布局文件控制 UI 界面</string >
    < string name = "app_name"> li3_1UI </string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
    < string name = "start">开始游戏</string >
</resources >
```

strings. xml 文件用于定义程序中应用的字符串常量。其中，每一个＜string＞子元素都可以定义一个字符串常量，常量名称由 name 属性指定，常量内容写在起始标记＜string＞和结束标记之间＜/string＞。

（4）在主活动中，即 MainActivity 中，用以下代码指定活动应用的布局文件。

```
setContentView(R.layout.main);
```

运行程序，效果如图 3-1 所示。

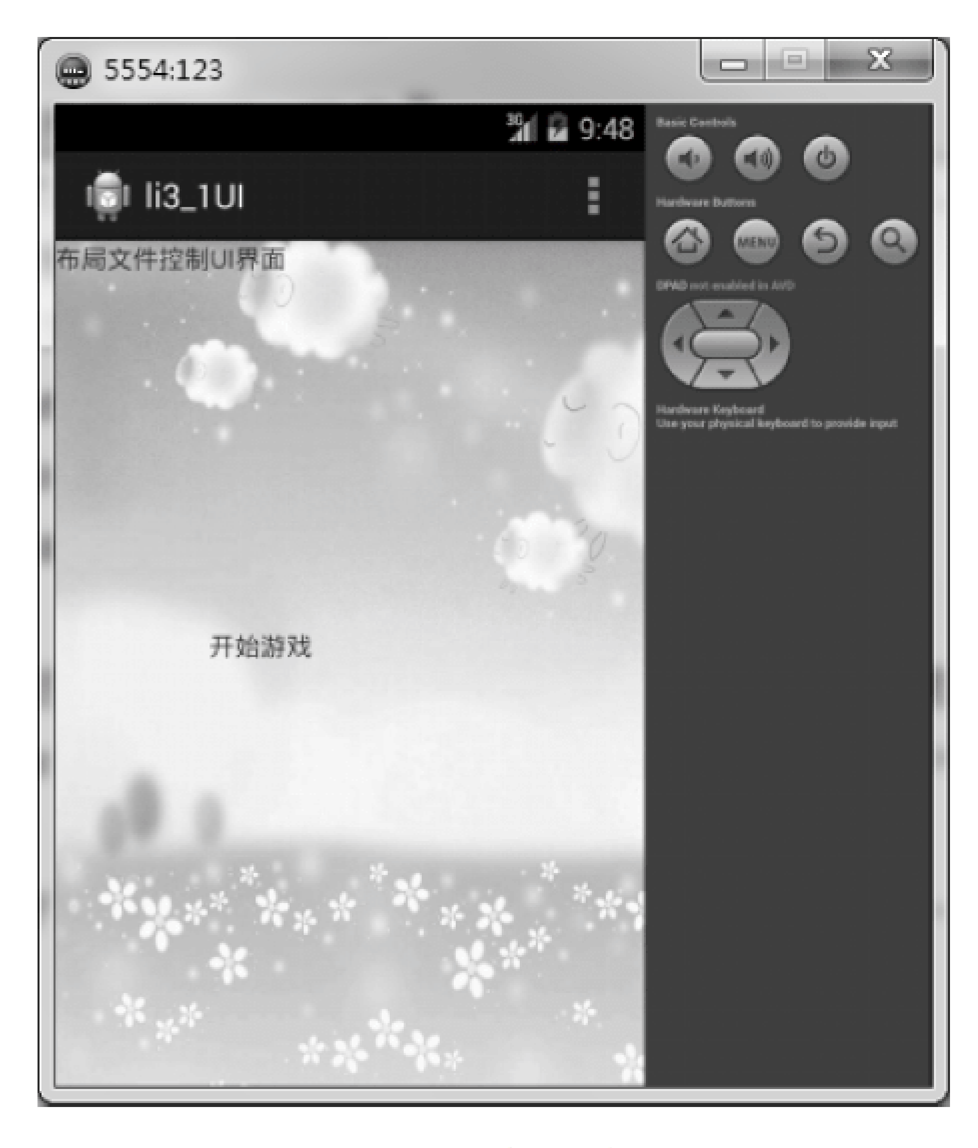


图 3-1　游戏开始界面

3.1.2 代码控制 UI

Android 支持像 Java Swing 那样完全通过代码控制 UI 界面。即将所有的 UI 控件都通过 new 关键字创建出来，然后将这些 UI 控件添加到布局管理器中，从而实现用户界面。

通过代码控制 UI 界面可分为以下几个步骤：

(1) 创建布局管理器，可以是帧布局管理器、表格布局管理器、线性布局管理器和相对布局管理器等，并且设置布局管理器的属性。例如，为布局管理器设置背景图片等。

(2) 创建具体的控件，可以是 TextView、ImageView、EditText 和 Button 等任何 Android 提供的控件，并且设置控件的布局和各种属性。

(3) 将创建的具体控件添加到布局管理器中。

【例 3-2】 完全通过代码实现游戏的进入界面。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li3_2UI。

(2) 在新创建的项目中打开 src\fs.li3_2UI 目录下的 MainActivity.java 文件：

```
public class MainActivity extends Activity
{
    //声明一个 TextView 控件 text2,为控件添加事件监听
    public TextView text2;
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
       //创建一个帧布局管理器,并为该布局管理器设置背景
        FrameLayout frameLayout = new FrameLayout(this);          //创建帧布局管理器
        frameLayout.setBackgroundDrawable(this.getResources().getDrawable(
                R.drawable.bj1));                          //背景
        setContentView(frameLayout);                       //在 Activity 中显示 frameLayout
        //创建一个 TextView 控件 text1,设置其文字大小和颜色,并将其添加到布局管理器中
        TextView text1 = new TextView(this);
        text1.setText("在代码中控制 UI 界面");          //显示的文字
        //设置文字大小,单位为像素
        text1.setTextSize(TypedValue.COMPLEX_UNIT_PX, 24);
        text1.setTextColor(Color.rgb(1, 1, 1));          //设置文字的颜色
        frameLayout.addView(text1);                        //将 text1 添加到布局管理器中
        //实例化 text2 控件,设置其文字、文字大小、颜色和布局
        text2 = new TextView(this);
        text2.setText("进入游戏......");                //显示文字
        //设置文字大小,单位为像素
        text2.setTextSize(TypedValue.COMPLEX_UNIT_PX, 24);
        text2.setTextColor(Color.rgb(1, 1, 1));          //设置文字的颜色
        LayoutParams params = new LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
        ViewGroup.LayoutParams.WRAP_CONTENT);             //创建保存布局参数的对象
        params.gravity = Gravity.CENTER_HORIZONTAL|Gravity.CENTER_VERTICAL;   //居中显示
        text2.setLayoutParams(params);                     //设置布局参数
        text2.setOnClickListener(new OnClickListener(){          //为 text2 添加单击事件监听
            @Override
            public void onClick(View v) {
    new AlertDialog.Builder(MainActivity.this).setTitle("系统提示")          //对话框的标题
```

```
.setMessage("进入游戏可能有风险,真的要进入吗?")   //设置对话框的显示内容
                        .setPositiveButton("确定",//为"确定"按钮添加单击事件
                                new DialogInterface.OnClickListener(){
                                    @Override
            public void onClick(DialogInterface dialog,
                    int which) {
            Log.i("3.2", "进入游戏");               //输出消息日志
            }
}).setNegativeButton("退出",                    //为"退出"按钮添加单击事件
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog,
                    int which) {
            Log.i("3.2", "退出游戏");               //输出消息日志
                finish();                           //结束游戏
            }
    }).show();
}
            });
          frameLayout.addView(text2);              //将 text2 添加到布局管理器中
        }
}
```

运行程序,效果如图 3-2 所示。单击"进入游戏……",将弹出如图 3-3 所示的提示对话框。

说明:完全通过代码控制 UI 界面虽然比较灵活,但是其开发过程比较烦琐,而且不利于高层次的解耦,因此不推荐使用这种方式控制 UI 界面。



图 3-2 游戏开始界面



图 3-3 显示提示对话框

3.1.3 混合控制 UI

完全通过 XML 布局文件控制 UI,虽然实现比较方便、快捷,但是有失灵活;完全通过 Java 代码控制 UI,虽然比较灵活,但是开发过程比较烦琐。比较这两种方法的优缺点,下面

介绍另一种控制 UI 的方法，即使用 XML 和 Java 代码混合控制 UI。

使用 XML 和 Java 代码混合控制 UI 界面，习惯上把变化较少、行为比较固定的组件放在 XML 布局文件中，把变化较多、行为控制较复杂的组件放在 Java 代码中。下面通过一个实例来演示怎样使用 XML 和 Java 代码混合控制 UI。

【例 3-3】 通过 XML 和 Java 代码在窗体中纵向显示 3 张图片。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li3_3UI。

(2) 打开 res\layout 目录下的布局 main.xml，将代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = " vertical"
    android:layout_width = "wrap_content"
    android:layout_height = "match_parent"
    android:background = "@drawable/bj1"
    android:id = "@ + id/layout" >
</LinearLayout >
```

(3) 在新创建的项目中打开 src\fs.li3_3UI 目录下的 MainActivity.java 文件：

```
public class MainActivity extends Activity {
    private ImageView[ ] a = new ImageView[3];          //声明一个保存 ImageView 控件的数组
    private int[ ] aPath = new int[ ]{
        R.drawable.a01,R.drawable.a02,R.drawable.a03
    };                                                  //声明并初始化一个保存访问图片的数组
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取 XML 文件中定义的线性布局管理器
        LinearLayout layout = (LinearLayout)findViewById(R.id.layout);
        for(int i = 0;i < aPath.length;i++){
            a[i] = new ImageView(this);                 //创建一个 ImageView 控件
            a[i].setImageResource(aPath[i]);            //为 ImageView 控件指定要显示的图片
            a[i].setPadding(5, 5, 5, 5);                //设置 ImageView 控件的内边距
            //设置图片的宽度和高度
            LayoutParams params = new LayoutParams(254,168);
            a[i].setLayoutParams(params);               //为 ImageView 控件设置布局参数
            layout.addView(a[i]);                       //将 ImageView 控件添加到布局管理器中
        }
    }
}
```

(4) 打开 AndroidManifest.xml 文件，将代码修改为：

```
< uses - sdk
    android:minSdkVersion = "8"
    android:targetSdkVersion = "18" />
< uses - permission android:name = "android.permission.SET_WALLPAPER"/>
```

运行程序，效果如图 3-4 所示。



图 3-4　纵向显示 3 张图片

3.2　View 对象

3.2.1　View 概述

在介绍 Android 的布局管理器之前，有必要让读者了解 Android 平台下的控件类。首先要了解的是 View 类，该类为所有可视化控件的基类，主要提供了控件绘制和事件处理的方法。创建用户界面所使用的控件都继承自 View，例如 TextView、Button、CheckBox 等。

关于 View 及其子类的相关属性，既可以在 XML 布局文件中进行设置，也可以通过成员方法在代码中动态设置。View 类常用的属性及其对应方法如表 3-1 所示。

表 3-1　View 类常用的属性及对应方法说明

属性名称	对应方法	描　述
android:background	setBackgroundResource(int)	设置背景
android:clickable	setClickable(boolean)	设置界面单击效果
android:visibility	setVisibility(int)	控制界面可见性
android:focusable	setFocusable(boolean)	控制界面可聚焦
android:id	setId(int)	界面索引号
android:longClickable	setLongClickable(boolean)	设置界面长单击效果
android:soundEffectsEnabled	setSoundEffectsEnabled(boolean)	设置音效启动效果
android:saveEnabled	setSaveEnabled(boolean)	设置界面除启动性能

续表

属性名称	对应方法	描述
android:nextFocusDown	setNextFocusDownId(int)	定义当向下搜索时应该获取焦点的 View，如果该 View 不存在或者不可见，则会抛出 RuntimeException 异常
android:nextFocusLeft	setNextFocusLeftId(int)	定义当向左搜索时应该获取焦点的 View
android:nextFocusRight	setNextFocusRightId(int)	定义当向右搜索时应该获取焦点的 View
android:nextFocusUp	setNextFocusUpId(int)	定义当向上搜索时应该获取焦点的 View，如果该 View 不存在或者不可见，则会抛出 RuntimeException 异常

说明：任何继承自 View 的子类都拥有 View 类的以上属性及对应方法。

3.2.2 ViewGroup 概述

另外一个需要了解的是 ViewGroup 类，它也是 View 类的子类，但是可以充当其他控件的容器。ViewGroup 的子控件既可以是普通的 View，也可以是 ViewGroup，实际上使用了 Composite 的设计模式。Android 中的一些高级控件(如 Galley、GridView 等)都继承自 ViewGroup。

与 Java SE 不同，在 Android 中并没有设计布局管理器，而是为每种不同的布局提供了一个 ViewGroup 的子类，常用的布局及其类结构如图 3-5 所示。

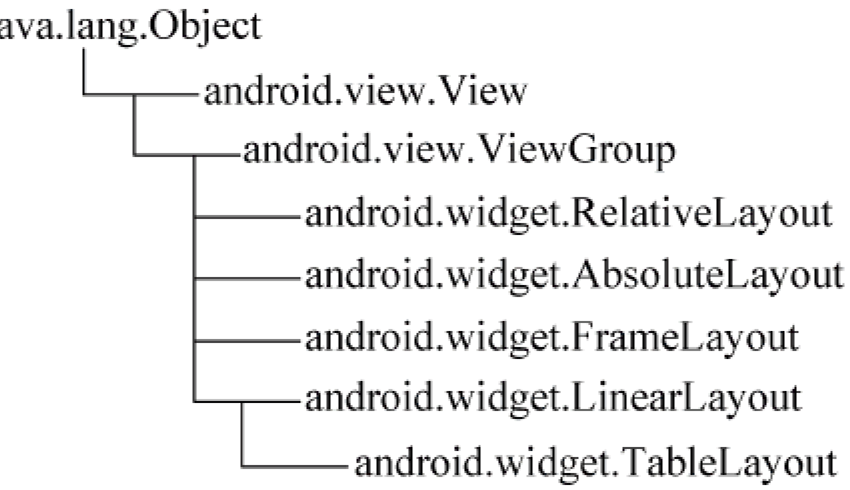


图 3-5 布局管理器的类结构

3.2.3 自定义 View

虽然 Android 提供了很多继承了 View 类的 UI 控件，但是在实际开发时还会出现不足以满足程序需要的情况，这时用户可以通过继承 View 类来开发自己的控件。开发自定义的 View 控件主要有以下步骤：

(1) 创建一个继承 android. view. View 类的 View 类，并且重写构造方法。

(2) 根据需要重写相应的方法。

在代码中右击，在弹出的快捷菜单中选择“代码→覆盖/实现方法”命令，将弹出如图 3-6 所示的对话框，在该对话框的列表框中显示出了可以被重写的方法，选中要重写方法前面的复选框，单击“确定”按钮，Eclipse 将自动重写指定的方法。一般情况下，不需要重写全部方法。

(3) 在项目的活动中，创建并实例化自定义 View 类，然后将其添加到布局管理器中。

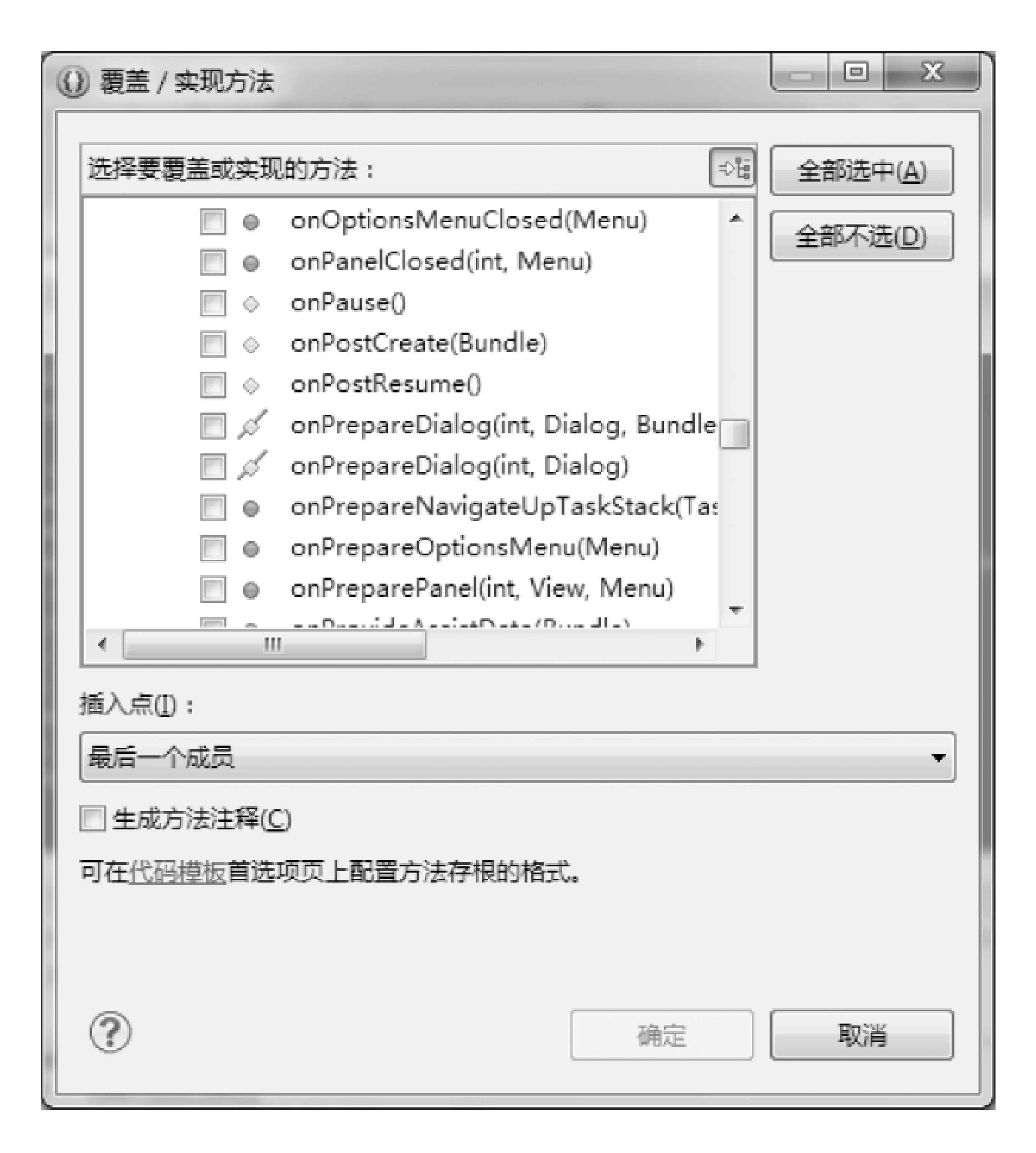


图 3-6 “覆盖/实现方法”对话框

3.2.4 View 对象实例

下面通过一个具体的实例来演示怎样开发自定义 View 类。

【例 3-4】 自定义 View 控件实现跟随手指动的小鸭子。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li3_4View。

(2) 打开 res\layout 目录下的 main.xml 布局文件，将其代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:background = "@drawable/bj1"
    android:id = "@ + id/mylayout" >
</FrameLayout >
```

(3) 创建一个名为 DcukView 的 Java 类，该类继承 android.view.View 类，重写带一个参数 Context 的构造方法和 onDraw()方法。其中，在构造方法中设置小鸭子的默认显示位置，在 onDraw()方法中根据图片绘制小鸭子：

```
public class DcukView extends View {
    public float bitmapX;                    //小鸭子显示位置的 X 坐标
    public float bitmapY;                    //小鸭子显示位置的 Y 坐标
    public DcukView(Context context) {       //重写构造方法
```

```
        super(context);
        bitmapX = 750;                          //小鸭子的默认显示位置的 X 坐标
        bitmapY = 500;                          //小鸭子的默认显示位置的 Y 坐标
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();              //创建并实例化 Paint 的对象
        Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(),
                R.drawable.duck1);              //根据图片生成位图对象
        canvas.drawBitmap(bitmap, bitmapX, bitmapY, paint);   //绘制小小鸭
        if (bitmap.isRecycled()) {              //判断图片是否回收
            bitmap.recycle();                   //强制回收图片
        }
    }
}
```

(4) 打开主活动文件 MainActivity,在它的 onCreate()方法中首先获取帧布局管理器并实例化小鸭子对象 duck,接着为 duck 添加触摸事件监听器,在重写的触摸事件中设置 duck 的显示位置并重绘 duck 控件,最后将 duck 添加到布局管理器中:

```
public class MainActivity extends Activity {
    /* 第一次调用 Activity */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取帧布局管理器
        FrameLayout frameLayout = (FrameLayout) findViewById(R.id.mylayout);
        //创建并实例化 RabbitView 类
        final DcukView rabbit = new DcukView(MainActivity.this);
        //为小鸭子添加触摸事件监听
        rabbit.setOnTouchListener(new OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                rabbit.bitmapX = event.getX();      //小鸭子显示位置的 X 坐标
                rabbit.bitmapY = event.getY();      //小鸭子显示位置的 Y 坐标
                rabbit.invalidate();                //重绘 rabbit 控件
                return true;
            }
        });
        frameLayout.addView(rabbit);                //将 rabbit 添加到布局管理器中
    }
}
```

运行程序,效果如图 3-7 所示,当用鼠标在屏幕上拖动时,小鸭子将跟随鼠标的拖动轨迹移动,效果如图 3-8 所示。



图 3-7　主界面

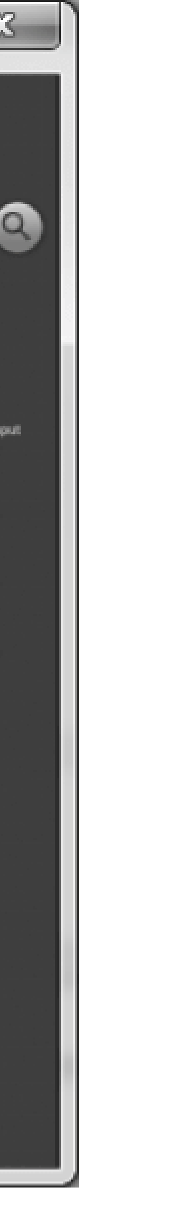


图 3-8　随鼠标拖动的效果图

3.3　布局管理器

在 Android 中，每个控件在窗体中都有具体的位置和大小，在窗体中摆放各种控件时，很难判断其具体位置和大小。不过，使用 Android 布局管理器可以很方便地控制各控件的位置和大小。Android 提供了线性布局(LinearLayout)、表格布局(TableLayout)、帧布局(FrameLayout)、相对布局(RelativeLayout)和绝对布局(AbsoluteLayout)5 种布局管理器。对应这 5 种布局管理器，Android 提供了 5 种布局方式，其中，绝对布局在 Android 2.0 中被标记为已过期。

3.3.1　线性布局

线性布局是将放入其中的控件按照垂直或水平方向来布局，也就是控制放入其中的控件横向排列或纵向排列。在线性布局中，每一行(针对垂直排列)或每一列(针对水平排列)中只能放一个控件，并且 Android 的线性布局不会换行，当控件一个挨着一个排列到窗体的边缘后，剩下的控件将不会被显示出来。

在线性布局中，排列方式由 android:orientation 属性来控制，对齐方式由 android:gravity 属性来控制。

在 Android 中，可以在 XML 布局文件中定义线性布局管理器，也可以使用 Java 代码来创建。推荐在 XML 布局文件中定义线性布局管理器。在 XML 布局文件中定义线性布局管理器，需要使用<LinearLayout>标记，其格式为：

```
<LinearLayout xmlns: android = "http://schemas.android.com/apk/res/android"
  属性列表>
```



```
</LinearLayout>
```

在线性布局管理器中，常用的属性有 android:orientation、android:gravity、android:layout_width、android:layout_height、android:id 和 android:background。其中，前两个属性是线性布局管理器支持的属性，后面 4 个为 android.view.View 和 android.view.ViewGroup 支持的属性，说明如下。

- android:orientation 属性：该属性用于设置布局管理器内控件的排列方式，其可选值有 horizontal 和 vertical，默认值为 vertical。其中，horizontal 表示水平排列，vertical 表示垂直排列。
- android:gravity 属性：该属性用于设置布局管理器内控件的对齐方式，其可选值有 top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 和 clip_horizontal。这些属性值也可以同时指定，各属性值之间用竖线隔开。例如要指定控件靠右下角对齐，可以使用属性值 right|bottom。
- android:layout_width 属性：该属性用于设置控件的基本宽度，其可选值有 fill_parent、match_parent 和 wrap_content。其中，fill_parent 表示控件的宽度与父容器的宽度相同；match_parent 和 fill_parent 的作用完全相同，从 Android 2.2 开始推荐使用；wrap_content 表示控件的宽度恰好能包括它的内容。

说明：android:layout_width 属性为 ViewGroup.LayoutParams 所支持的 XML 属性，对于其他的布局管理器同样适用。

- android:layout_height 属性：该属性用于设置控件的基本高度，其可选值有 fill_parent、match_parent 和 wrap_content。其中，fill_parent 表示控件的高度与父容器的高度相同；match_parent 和 fill_parent 的作用完全相同，从 Android 2.2 开始推荐使用；wrap_content 表示控件的高度恰好能包括它的内容。

说明：android:layout_height 属性是 ViewGroup.LayoutParams 所支持的 XML 属性，对于其他的布局管理器同样适用。

- android:id 属性：该属性用于为当前控件指定一个 ID 属性，在 Java 代码中可以应用该属性单独引用这个控件。为控件指定 ID 属性后，在 R.java 文件中会自动派生一个对应的属性，在 Java 代码中，可以通过 findViewById()方法获取它。
- android:background 属性：该属性用于为控件设置背景，可以是背景图片，也可以是背景颜色。在为控件指定背景图片时，可以将准备好的背景图片复制到目录下，然后使用以下代码设置：

```
android:background = "@drawable/background"
```

如果想指定背景颜色，可以使用颜色值。例如，想指定背景颜色为白色，可以使用下面的代码：

```
android:background = "#FFFFFFFF"
```

说明：在线性布局中，还可以使用 android.view.View 类支持的其他属性。

【例 3-5】 线性布局管理实例。

其实现步骤如下：

(1) 在 Eclipse 中新建一个项目 li3_5LinearLayout，首先打开项目文件夹中 res\values 目录下的 strings. xml，在其中输入以下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "hello"> Example1 </string >
    < string name = "app_name"> Example1 </string >
</resources >
```

说明：在 strings. xml 中主要声明了程序中要用到的字符串资源，将所有字符串资源统一管理有助于提高程序的可读性及可维护性。

(2) 打开项目文件夹中 res\layout 目录下的 main. xml，将其中已有的代码替换为以下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
              android:layout_width = "fill_parent"
              android:layout_height = "fill_parent"
              android:orientation = "horizontal">   <! -- 设置线性布局为水平方向 -->
    < Button android:id = "@ + id/but1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "but1"
            android:layout_weight = "1" />
    < Button android:id = "@ + id/but1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "but2"
    android:layout_weight = "1" />
    < Button android:id = "@ + id/but3"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "but3"
    android:layout_weight = "1" />
    < Button android:id = "@ + id/but4"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "but4"
    android:layout_weight = "1" />
    < Button android:id = "@ + id/but5"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "but5"
    android:layout_weight = "1" />
</LinearLayout >
```

运行程序，效果如图 3-9 所示。

在以上代码中，在 LinearLayout 视图组(ViewGroup)中包含了 5 个 Button，它的子元素是以线性方式(horizontal，水平的)布局的。

如果将代码中的 android: orientation ＝ " horizontal" 修改为 android: orientation ＝ "vertical"，即得到如图 3-10 所示的效果。

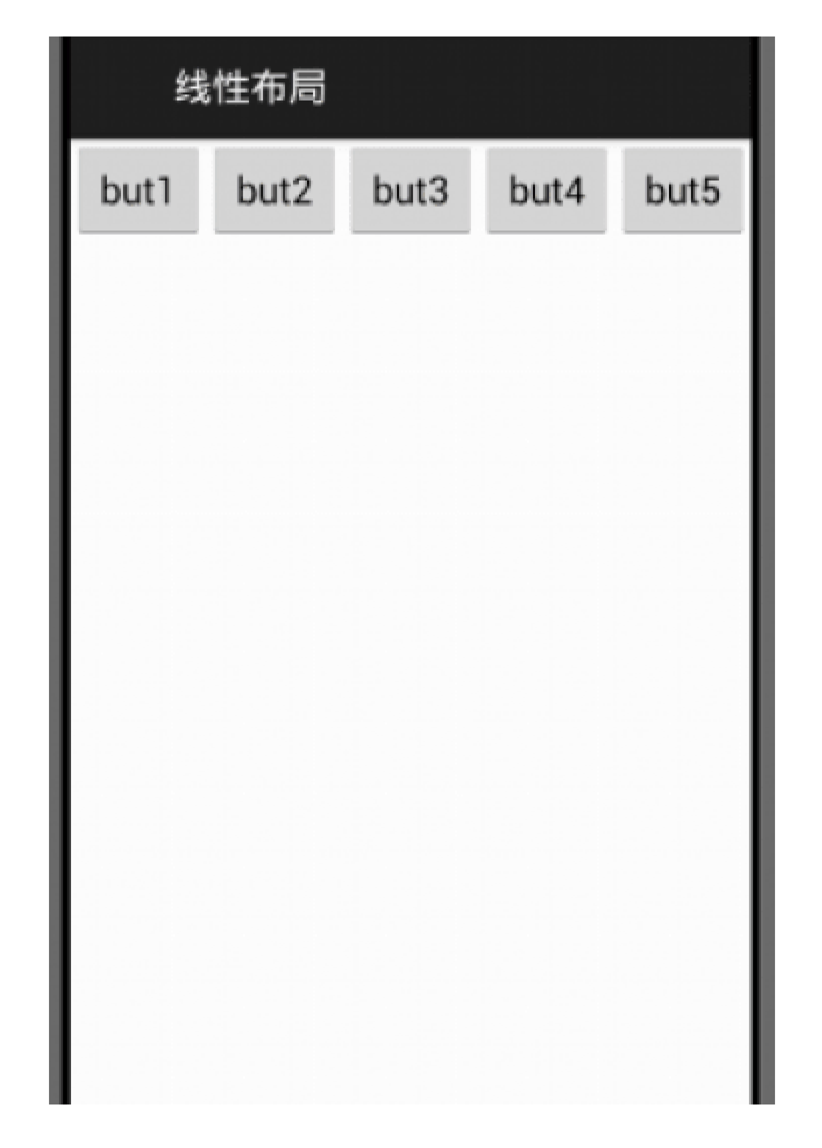


图 3-9　横向线性布局效果图

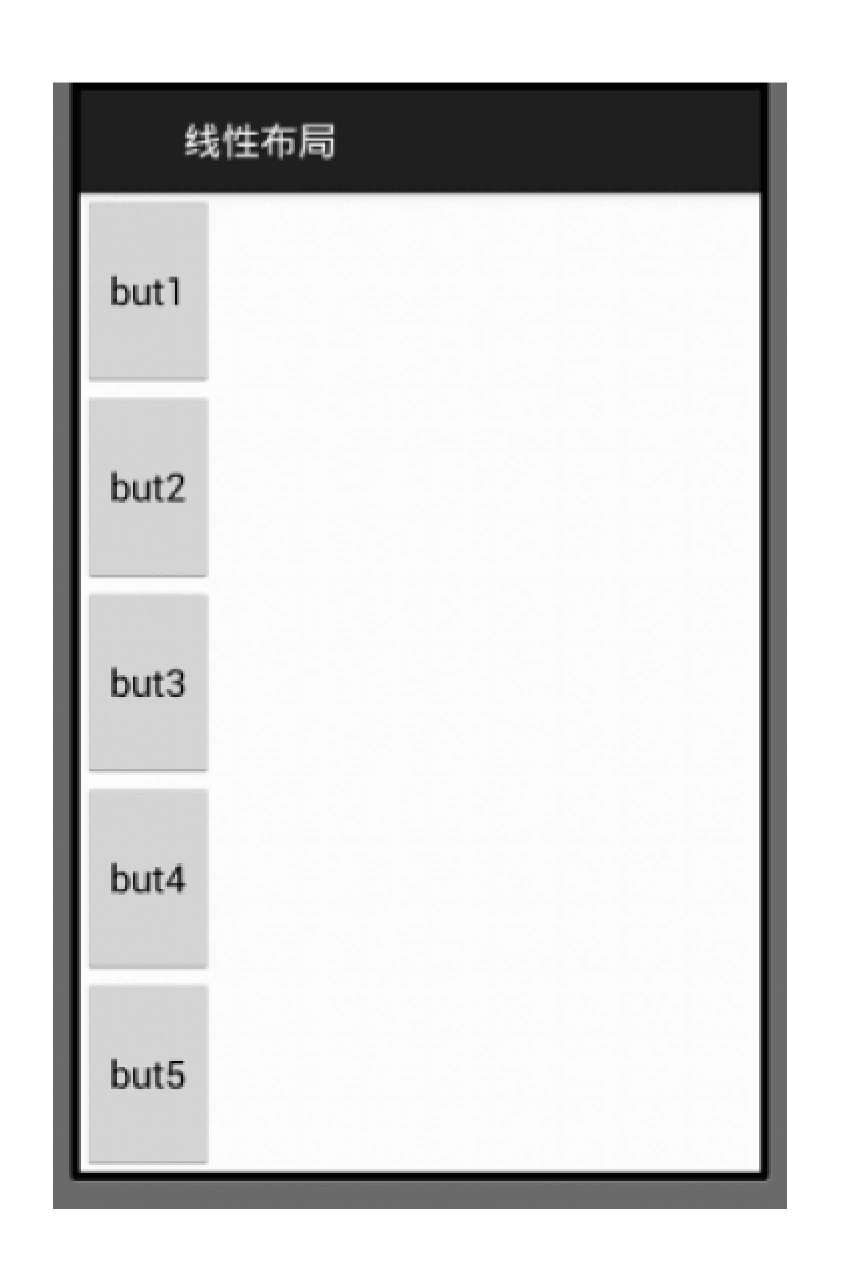


图 3-10　竖向线性布局效果图

3.3.2　表格布局

表格布局与常见的表格类似，它以行、列的形式来管理放入其中的 UI 控件。表格布局使用<TableLayout>标记定义，在表格布局中，可以添加多个<TableRow>标记，每个<TableRow>标记占用一行，由于<TableRow>标记也是容器，所以在该标记中还可以添加其他组件。在<TableRow>标记中，每添加一个控件，表格就会增加一列。在表格布局中，列可以被隐藏，也可以被设置为伸展，从而填充可利用的屏幕空间，还可以被设置为强制收缩，直到表格匹配屏幕大小。

说明：如果在表格布局中直接向<TableLayout>中添加 UI 控件，那么这个控件将独占一行。

在 Android 中，可以在 XML 布局文件中定义表格布局管理器，也可以使用 Java 代码来创建。推荐在 XML 布局文件中定义表格布局管理器。在 XML 布局文件中定义表格布局管理器的格式为：

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    属性列表>
    <TableRow 属性列表> 需要添加的 UI 控件 </TableRow>
     多个<TableRow>
</TableLayout>
```

TableLayout 继承了 LinearLayout，因此它完全支持 LinearLayout 所支持的全部 XML 属性。此外，TableLayout 支持如表 3-2 所示的 XML 属性。

表 3-2　TableLayout 支持的 XML 属性

XML 属性	描　述
android:collapseColumns	设置需要被隐藏的列序号(序号从 0 开始),多个列序号之间用逗号“,”分隔
android:shrinkColumns	设置允许被收缩的列的列序号(序号从 0 开始),多个列序号之间用逗号“,”分隔
android:stretchColumns	设置允许被拉伸的列的列序号(序号从 0 开始),多个列序号之间用逗号“,”分隔

【例 3-6】 表格布局管理实例。

其实现步骤如下:

(1) 使用 Eclipse 创建一个名为 li3_6TableLayout 的 Android 应用项目。

(2) 打开项目 res\values 目录下的 strings. xml,在其中输入以下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "hello"> Example2 </string >
    < string name = "app_name"> Example2 </string >
</resources >
```

(3) 打开项目 res\layout 目录下的 main. xml 文件,将其中已有的代码替换为以下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
    < TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent" >
        < TableRow
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:padding = "14dip">
            < TextView
                android:text = "姓名"
                android:gravity = "left" />
            < TextView
                android:text = "电话"
                android:gravity = "right"/>
        </TableRow >
        < View
            android:layout_height = "2dip"
            android:background = "#FFFFFF" />
        < TableRow
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:padding = "14dip">
            < TextView
                android:text = "AA"
                android:gravity = "left" />
            < TextView
            android:text = "000 - 555 - 111"
```

```
                android:gravity = "right"/>
        </TableRow >
        < TableRow
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:padding = "14dip">
            < TextView
                android:text = "BB"
                android:gravity = "left" />
            < TextView
                android:text = "222 - 000 - 333"
                android:gravity = "right"/>
        </TableRow >
        < TableRow
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:padding = "14dip">
            < TextView
                android:text = "AB"
                android:gravity = "left" />
            < TextView
                android:text = "222 - 333 - 111"
                android:gravity = "right"/>
        </TableRow >
        < TableRow
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:padding = "14dip" >
            < TextView
                android:text = "姓名"
                android:gravity = "left" />
            < TextView
                android:text = "性别"
                android:gravity = "right"/>
        </TableRow >
        < View
            android:layout_height = "2dip"
            android:background = "#FFFFFF" />
        < TableRow
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:padding = "14dip" >
            < TextView
                android:text = "AA"
                android:gravity = "left" />
            < TextView
                android:text = "女"
                android:gravity = "right"/>
        </TableRow >
        < TableRow
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:padding = "14dip">
            < TextView
```

```
                android:text = "BB"
                android:gravity = "left" />
            < TextView
                android:text = "男"
                android:gravity = "right"/>
        </TableRow >
        < TableRow
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:padding = "14dip">
            < TextView
                android:text = "AB"
                android:gravity = "left"/>
      < TextView
                android:text = "男"
                android:gravity = "right"/>
        </TableRow >
    </TableLayout >
```

运行程序，效果如图 3-11 所示。

图 3-11 表格布局

3.3.3 帧布局

在帧布局管理器中，每加入一个控件都将创建一个空白的区域，通常称为一帧，这些帧会根据 gravity 属性执行自动对齐。默认情况下，帧布局从屏幕的左上角(0,0)坐标点开始布局，多个控件层叠排序，后面的控件覆盖前面的控件。

在 Android 中，可以在 XML 布局文件中定义帧布局管理器，也可以使用 Java 代码来创建。推荐在 XML 布局文件中定义帧布局管理器。在 XML 布局文件中定义帧布局管理器可以使用<FrameLayout>标记，格式为：

```
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    属性列表>
</FrameLayout >
```

FrameLayout 支持的常用 XML 属性如表 3-3 所示。

表 3-3　FrameLayout 支持的常用 XML 属性

XML 属性	描　　述
android:foreground	设置该帧布局容器的前景图像
android:foregroundGravity	定义绘制前景图像的 gravity 属性，即前景图像显示的位置

【例 3-7】 帧布局管理实例。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li3_7FrameLayout。

(2) 打开项目 res\values 目录下的 strings.xml，在其中输入以下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name">帧布局管理实例</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
</resources >
```

(3) 打开该项目下的 res\main.xml 文件夹，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout
     android:id = "@ + id/frameLayout1"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent"
     xmlns:android = "http://schemas.android.com/apk/res/android"
     android:foreground = "@drawable/face1"
     android:background = "@drawable/bj2"
     android:foregroundGravity = "bottom|right">
     <!-- 添加居中显示的红色背景的 TextView,将显示在最下层 -->
    < TextView android:text = "红色背景的 TextView"
     android:id = "@ + id/textView1"
     android:background = "#FFFF0000"
     android:layout_gravity = "center"
     android:layout_width = "300px"
     android:layout_height = "300px"/>
     <!-- 添加居中显示的橙色背景的 TextView,将显示在中间层 -- >
    < TextView android:text = "橙色背景的 TextView"
    android:id = "@ + id/textView2"
    android:layout_width = "200px"
    android:layout_height = "200px"
    android:background = "#FFFF6600"
```

```
    android:layout_gravity="center"/>
    <!-- 添加居中显示的黄色背景的 TextView,将显示在最上层 -->
    <TextView android:text="黄色背景的 TextView"
    android:id="@+id/textView3"
    android:layout_width="100px"
    android:layout_height="100px"
    android:background="#FFFFEE00"
    android:layout_gravity="center"/>
</FrameLayout>
```

运行程序,效果如图 3-12 所示。



图 3-12 帧布局管理器实例

3.3.4 相对布局

相对布局(RelativeLayout)是指一个 ViewGroup 以相对位置显示它的子视图(View)元素,一个视图可以指定相对于它的兄弟视图的位置(例如在给定视图的左边或下面)或相对于 RelativeLayout 的特定区域的位置(例如底部对齐或中间偏左)。

相对布局是设计用户界面的有力工具,因为它消除了嵌套视图组。如果用户发现使用了多个嵌套的 LinearLayout 视图组,可以考虑使用一个 RelativeLayout 视图组。

在 RelativeLayout 中元素具有的一些重要的属性。

(1) 属性值必须为 ID 的引用名,例如"@+id/button1"。

- android:layout_below:设置该元素在某元素的下方。
- android:layout_above:设置该元素在某元素的上方。
- android:layout_toLeftOf:设置该元素在某元素的左边。
- android:layout_toRightOf:设置该元素在某元素的右边。
- android:layout_alignTop:设置该元素的上边缘和某元素的上边缘对齐。
- android:layout_alignLeft:设置该元素的左边缘和某元素的左边缘对齐。

- android:layout_alignBottom：设置该元素的下边缘和某元素的下边缘对齐。
- android:layout_alignRight：设置该元素的右边缘和某元素的右边缘对齐。

（2）属性值为 true 或 false。

- android:layout_centerHorizontal：设置是否相对于父元素水平居中。
- android:layout_centerVertical：设置是否相对于父元素垂直居中。
- android:layout_centerInparent：设置是否相对于父元素完全居中。
- android:layout_alignParentBottom：设置是否紧靠父元素的下边缘。
- android:layout_alignParentLeft：设置是否紧靠父元素的左边缘。
- android:layout_alignParentRight：设置是否紧靠父元素的右边缘。
- android:layout_alignParentTop：设置是否紧靠父元素的上边缘。
- android:layout_alignWithParentIfMissing：设置如果 layout_toLeftOf、layout_toRightOf 对应的元素找不到是否以父元素做参照。

【例 3-8】 相对布局实例。

其实现步骤如下：

（1）在 Eclipse 中创建一个名为 li3_8RelativeLayout 的 Android 应用项目。

（2）打开 res\layout 目录下的 main.xml 文件，代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@drawable/bj1"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/lable"
        android:text="类型为:"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/lable" />
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="确定"
        android:layout_below="@id/entry"
        android:layout_marginLeft="10px"
        android:layout_alignParentRight="true" />
    <Button
        android:id="@+id/cancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_toLeftOf = "@id/ok"
        android:layout_alignTop = "@id/ok"
        android:text = "取消" />
</RelativeLayout>
```

运行程序,效果如图 3-13 所示。



图 3-13　相对布局实例

相对布局容器中的子控件总是相对其他控件来决定分布位置,可以考虑先把一个控件放在相对布局容器的中间,然后以该控件为中心,将其他控件分布在该控件的四周,这样即可形成“烟花布局”效果。

【例 3-9】 “烟花”布局效果的界面布局文件代码如下。

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    android:background = "@drawable/bj1"
    tools:context = ".MainActivity" >
<!-- 定义该控件位于父容器中间 -->
<TextView
    android:id = "@ + id/view01"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/flow "
    android:layout_centerInParent = "true"/>
<!-- 定义该控件位于 view01 控件的上方 -->
<TextView
    android:id = "@ + id/view02"
```

```
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/flow "
    android:layout_above = "@id/view01"
    android:layout_alignLeft = "@id/view01"/>
<! -- 定义该控件位于 view01 控件的下方 -->
< TextView
    android:id = "@ + id/view03"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/flow "
    android:layout_below = "@id/view01"
    android:layout_alignLeft = "@id/view01"/>
<! -- 定义该控件位于 view01 控件的左边 -->
< TextView
    android:id = "@ + id/view04"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/flow "
    android:layout_toLeftOf = "@id/view01"
    android:layout_alignTop = "@id/view01"/>
<! -- 定义该控件位于 view01 控件的右边 -->
< TextView
    android:id = "@ + id/view05"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/flow "
    android:layout_toRightOf = "@id/view01"
    android:layout_alignTop = "@id/view01"/>
</RelativeLayout >
```

运行程序,效果如图 3-14 所示。



图 3-14　烟花布局效果

3.4 选 项 卡

选项卡(TabWidget)是一种相对复杂的布局管理器,通过多个标签来切换显示不同的内容,一个 TabWidget 主要是由一个 TabHost 来存放多个 Tab 标签容器,再在 Tab 容器中加入其他控件,通过 addTab 方法可以增加新的 Tab,这些除了在 XML 文件中布局好控制外,当然还需要在 Java 文件中处理好事件的逻辑。

TabWidget 继承自 LinearLayout,是线性布局的一种,除了继承父类的属性和方法,在 FrameLayout 类中包含了自己特有的属性和方法,如表 3-4 所示。

表 3-4 TabWidget 常用的属性

属 性	描 述
android:divider	可绘制对象,被绘制在选项卡窗口间充当分割物
android:tabStripEnabled	确定是否在选项卡中绘制
android:tabStripLeft	被用来绘制选项卡下面的分割线左边部分的可视化对象
android:tabStripRight	被用来绘制选项卡下面的分割线右边部分的可视化对象

【例 3-10】 使用选项卡实例。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_10TabWidget。

(2) 打开 res\layout 目录下的 main.xml 文件,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义了 TabHost 布局,其 id 必须为 @android:id/tabhost -- >
< TabHost xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@android:id/tabhost"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <! -- 定义了 3 个选项卡的整体布局方式 -- >
    < LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
         <! -- 定义了选项卡 TabWidget,其 id 必须为@android:id/tabs -- >
        < TabWidget
            android:id = "@android:id/tabs"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <! -- 定义了选项卡的 FrameLayout 布局,其 id 必须为@android:id/tabcontent -- >
        < FrameLayout
            android:id = "@android:id/tabcontent"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent">
            < TextView
                android:id = "@ + id/textview1"
                android:layout_width = "fill_parent"
                android:layout_height = "fill_parent"
                android:text = "这是一个 tab" />
```

```
                <TextView
                    android:id="@+id/textview2"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:text="这是另一个 tab" />
                <TextView
                    android:id="@+id/textview3"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:text="这是第三个 tab" />
            </FrameLayout>
        </LinearLayout>
</TabHost>
```

(3) 打开 src\fs.li3_10TabWidget 目录下的 MainActivity 文件,其代码为:

```
public class MainActivity extends TabActivity
{
    //声明 TabHost 对象
    TabHost mTabHost;
    /* 第一次调用 Activity   */
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
         //取得 TabHost 对象
         mTabHost = getTabHost();
         /* 为 TabHost 添加标签 */
        //新建一个 newTabSpec(newTabSpec)
         //设置其标签和图标(setIndicator)
         //设置内容(setContent)
         mTabHost.addTab(mTabHost.newTabSpec("tab_test1")
             .setIndicator("TAB 1",getResources().getDrawable(R.drawable.b))
             .setContent(R.id.textview1));
         mTabHost.addTab(mTabHost.newTabSpec("tab_test2")
             .setIndicator("TAB 2",getResources().getDrawable(R.drawable.b1))
             .setContent(R.id.textview2));
         mTabHost.addTab(mTabHost.newTabSpec("tab_test3")
             .setIndicator("TAB 3",getResources().getDrawable(R.drawable.b2))
             .setContent(R.id.textview3));
         //设置 TabHost 的背景颜色
         mTabHost.setBackgroundColor(Color.argb(150, 22, 70, 150));
         //设置 TabHost 的背景图片资源
         mTabHost.setBackgroundResource(R.drawable.bj1);
         //设置当前显示哪一个标签
         mTabHost.setCurrentTab(0);
         //标签切换事件处理:setOnTabChangedListener
         mTabHost.setOnTabChangedListener(new OnTabChangeListener()
        {
             @Override
            public void onTabChanged(String tabId)
            {
                 Dialog dialog = new AlertDialog.Builder(MainActivity.this)
```

```
                    .setTitle("提示")
                    .setMessage("当前选中: " + tabId + "标签")
                    .setPositiveButton("确定", new DialogInterface.OnClickListener()
                    {
                    public void onClick(DialogInterface dialog, int whichButton)
                   {
                        dialog.cancel();
                   }
                }).create();                      //创建按钮
                dialog.show();
            }
        });
    }  }
```

运行程序,效果如图 3-15 所示。



图 3-15　选项卡界面

3.5　TabHost 容器

盛放 Tab 的容器就是 TabHost。TabHost 的实现有下面两种方式:

(1) 继承 TabActivity,从 TabActivity 中用 getTabHost()方法获取 TabHost。各个 Tab 中的内容在布局文件中定义就可以了。

(2) 不继承 TabActivity,在布局文件中定义 TabHost 即可,但是 TabWidget 的 ID 必须是@android:id/tabs,FrameLayout 的 ID 必须是@android:id/tabcontent。

【例 3-11】 只有一个 Activity 的简单例子,但要注意继承 tabActivity 这个类。

第 1 种方法:带 FrameLayout 布局。其具体实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 TabHost_1。

(2) 打开 res\layout 目录下的布局文件 main. xml,修改代码如下。注意这里用了

FrameLayout，主要特点是可以覆盖其他内容，只显示当前的 View Code。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/FrameLayout01"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/bj1">
    < TextView
        android:id = "@ + id/TextView01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "所有通话记录"></TextView >
    < TextView
        android:id = "@ + id/TextView02"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "已接来电"></TextView >
    < TextView
        android:id = "@ + id/TextView03"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "未接来电"></TextView >
</FrameLayout >
```

(3) 打开 src\fs.tabhost_1 目录下的 MainActivity 文件，将代码修改为：

```
public class MainActivity extends TabActivity {
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        TabHost th  =  getTabHost();
        //声明 TabHost,然后用 LayoutInflater 过滤出布局,给 TabHost 加上含有 Tab 页面
        //的 FrameLayout
        //from(this)从这个 TabActivity 获取 LayoutInflater
        //R.layout.main 存放 Tab 布局
        //通过 TabHost 获取存放 Tab 标签页内容的 FrameLayout
        //是否将 inflate 拴系到根布局元素上
        LayoutInflater.from(this).inflate(R.layout.main, th.getTabContentView(), true);
        //通过 TabHost 获取存放 Tab 标签页内容的 FrameLayout
        //newTabSpecd 的作用是获取一个新的 TabHost.TabSpec,并关联到当前 TabHost
        //setIndicator 的作用是指定标签和图标作为选项卡的指示符
        //setContent 的作用是指定用于显示选项卡内容的视图 ID
        th.addTab(th.newTabSpec("all").setIndicator("所有通话记录", getResources().
getDrawable(R.drawable.b)).setContent(R.id.TextView01));
        th.addTab(th.newTabSpec("ok").setIndicator("已接来电",getResources().getDrawable
(R.drawable.b1)).setContent(R.id.TextView02));
        th.addTab(th.newTabSpec("cancel").setIndicator("未接来电",getResources().
getDrawable(R.drawable.b2)).setContent(R.id.TextView03));
        //setOnTabChangeListener 的作用是注册一个回调函数,当任何一个选项卡的选中状态发
        //生改变时调用
        th.setOnTabChangedListener(
            new OnTabChangeListener() {
            @Override
                public void onTabChanged(String tabId) {
                Toast.makeText(MainActivity.this, tabId, Toast.LENGTH_LONG).show();
                }
```

```
            }
        );
    }
}
```

运行程序,效果如图 3-16 所示。



图 3-16 带 FrameLayout 布局的 TabHost 实例效果

第 2 种方法:没有用到 FrameLayout 这个布局,而是直接用了 listview,创建选项卡内容的回调函数 creatTabContent()实现。其具体实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 TabHost_2。

(2) 打开 res\layout 目录下的布局文件 main.xml,将代码修改为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:orientation = "vertical"
android:background = "@drawable/bj1">
< TextView
 android:layout_width = "fill_parent"
android:layout_height = "wrap_content"
 android:text = "@string/hello" />
 </LinearLayout >
```

(3) 打开 src\fs.tabhost_2 目录下的 MainActivity 文件,将代码修改为:

```
//TabContentFactory 为当某一选项卡被选中时生成选项卡的内容
//如果选项卡的内容按某些条件生成,请使用该接口,例如不显示既存的视图而是启动活动
public class MainActivity extends TabActivity implements TabHost.TabContentFactory {
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
```

```
            TabHost th = getTabHost();
            //newTabSpecd 的作用是获取一个新的 TabHost.TabSpec,并关联到当前 TabHost
            //setIndicator 的作用是指定标签和图标作为选项卡的指示符
            //setContent 的作用是指定用于显示选项卡内容的视图id
            th.addTab(th.newTabSpec("所有").setIndicator("所有通话记录").setContent(this));
            th.addTab(th.newTabSpec("确定").setIndicator("已接来电").setContent(this));
            th.addTab(th.newTabSpec("取消").setIndicator("未接来电").setContent(this));
        }
        //创建选项卡内容的回调函数
        public View createTabContent(String tag)
        {
            ListView lv = new ListView(this);
            List<String> list = new ArrayList<String>();
            list.add(tag);
            if(tag.equals("所有"))
            {
                list.add("Kate");
                list.add("Mimi");
                list.add("rose");
                }else if(tag.equals("ok"))
                {
                    list.add("Kate");
                    list.add("Mimi");
                    }else
                    {
                        list.add("rose");
                    }
                    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,android.R.
layout.simple_list_item_checked, list);
                    lv.setAdapter(adapter);
                    return lv;
            }
}
```

运行程序,效果如图 3-17 所示。



图 3-17 不带 FrameLayout 布局的 TabHost 实例效果

3.6 布局应用实例

前面已经对 Android 的布局类型做了介绍,下面通过实例来说明几个布局的应用。

【例 3-12】 霓虹灯效果实例。

如果考虑轮换改变上面帧布局中 7 个 TextView 的背景色,将会看到上面的颜色渐变不断地变换,就像大街上的霓虹灯一样。本实例使用 FrameLayout 布局管理器,只是程序启动了一个线程来周期性地改变这 7 个 TextView 的背景色。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_12Color。

(2) 打开 res\layout 目录下的 main.xml 布局文件,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1">
<!-- 依次定义 7 个 TextView,先定义的 TextView 位于底层,后定义的 TextView 位于上层 -->
<TextView android:id = "@ + id/View01"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:width = "210px"
    android:height = "50px"
    android:background = "#ff0000"/>
<TextView android:id = "@ + id/View02"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:width = "180px"
    android:height = "50px"
    android:background = "#dd0100"/>
<TextView android:id = "@ + id/View03"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:width = "150px"
    android:height = "50px"
    android:background = "#bb0001"/>
<TextView android:id = "@ + id/View04"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:width = "120px"
    android:height = "50px"
    android:background = "#980000"/>
<TextView android:id = "@ + id/View05"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:width = "90px"
    android:height = "50px"
    android:background = "#770000"/>
<TextView android:id = "@ + id/View06"
```

```
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "60px"
        android:height = "50px"
        android:background = "#550050"/>
    <TextView android:id = "@+id/View07"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "30px"
        android:height = "50px"
        android:background = "#332000"/>
    </FrameLayout>
```

(3) 打开 src\fs.li3_12color 目录下的 MainActivity 文件，在代码中定义一个每 0.1s 执行一次的任务，该任务仅仅改变 currentColor 变量的值，然后向 Handler 发送一条消息，通知它更新 7 个 TextView 的背景色。代码为：

```
public class MainActivity extends Activity
{
    private int currentColor = 0;
    //定义一个颜色数组
    final int[] colors = new int[]{
        R.color.color7,
        R.color.color6,
        R.color.color5,
        R.color.color4,
        R.color.color3,
        R.color.color2,
        R.color.color1,};
        final int[]
        names = new int[]{
            R.id.View01,
            R.id.View02,
            R.id.View03,
            R.id.View04,
            R.id.View05,
            R.id.View06,
            R.id.View07};
            TextView[] views = new TextView[7];
            @Override
            public void
            onCreate(Bundle savedInstanceState){
                super.onCreate(savedInstanceState);
                setContentView
                (R.layout.main);
                for (int i = 0 ; i < 7 ; i++){
                    views[i] = (TextView)findViewById
                    (names[i]);}
                    final Handler handler = new
                    Handler(){
                    @Override
                    public void handleMessage(Message msg){
                        //表明消息来自本程序
                        if(msg.what == 0x1122){
```

```
            //依次改变 7 个 TextView 的背景色
            for(int i = 0 ; i < 7 - currentColor ; i++){
        views[i].setBackgroundResource(colors[i + currentColor]);}
                for(int i = 7 - currentColor , j = 0 ; i < 7 ; i++,j++){
                    views[i].setBackgroundResource(colors[j]);}
                }
                super.handleMessage(msg);}
            };
            //定义一个线程周期性地改变 currentColor 变量的值
            new Timer().schedule(new TimerTask(){
                @Override
                public void run(){
                    currentColor++;
                    if(currentColor >= 6){
                        currentColor = 0;}
            //发送一条消息通知系统改变 7 个 TextView 控件的背景色
                        Message m = new Message();
                        //给该消息定义一个标识
                        m.what = 0x1122;
                        handler.sendMessage(m);
                    }
                }, 0 , 100);
        }
}
```

(4) 选择 res\values 目录并右击，在弹出的快捷菜单中选择"新建→文件"命令，创建一个名为 colors.xml 文件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <color name = "color1">#330000</color>
    <color name = "color2">#550000</color>
    <color name = "color3">#770000</color>
    <color name = "color4">#990000</color>
    <color name = "color5">#bb0000</color>
    <color name = "color6">#dd0000</color>
    <color name = "color7">#ff0000</color>
</resources>
```

运行程序，效果如图 3-18 所示。

在手机上浏览图片时，一般都是一屏只浏览一张图片，通过触摸事件可以改变显示的图片，下面通过一个实例来达到这个效果。

【例 3-13】 实现一个简易的图片浏览器。

本例实现一个简易的图片浏览器，即在窗体上显示一张图片，触摸该图片时将显示下一张图片，再次触摸还会切换一张图片，直到提供的图片全部显示后再从第一张图片开始显示。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 项目，命名为 li3_13Picture。

(2) 打开 res\layout 目录下的 main.xml 布局文件，将默认添加的布局删除，然后添加一个 LinearLayout 线性布局，并设置该布局管理器的背景、布局管理器内控件的对齐方式和 ID 属性。代码为：



图 3-18　霓虹灯效果

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1"
    android:gravity = "center"
    android:id = "@ + id/layout">
</LinearLayout >
```

(3) 打开 src\fs.li3_13picture 目录下的 MainActivity 文件，在文件中创建一个当前索引的整型成员变量和一个保存访问图片的数组。在 onCreate()方法中获取 XML 文件中定义的线性布局，然后创建一个 ImageView 控件，并设置该控件要显示的图片、宽度、高度、布局参数和触摸事件监听器，最后将 ImageView 控件添加到布局管理器中。代码为：

```
public class MainActivity extends Activity {
    private int index = 0;                          //当前索引
    private int[] imagePath = new int[]{
        R.drawable.a01,R.drawable.a04,R.drawable.a03,R.drawable.a02,R.drawable.a05
    };                                              //声明并初始化一个保存访问图片的数组
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
      //获取 XML 文件中定义的线性布局管理器
        LinearLayout layout = (LinearLayout)findViewById(R.id.layout);
            ImageView img = new ImageView(this);    //创建一个 ImageView 控件
      img.setImageResource(imagePath[index]);       //为 ImageView 控件指定要显示的图片
        LayoutParams params = new LayoutParams(253,148);           //设置图片的宽度和高度
            img.setLayoutParams(params);            //为 ImageView 控件设置布局参数
```

```
            img.setOnTouchListener(new OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                    if(index<3){
                        index++;
                    }else{
                        index=0;
                    }
                    //为 ImageView 控件指定要显示的图片
                    ((ImageView)v).setImageResource(imagePath[index]);
        return false;
                }
            });
        layout.addView(img);                    //将 ImageView 控件添加到布局管理器中
    }
}
```

说明：在为 ImageView 控件添加触摸事件监听器时，需要在重写的 OnTouch()事件中实现更改 ImageView 控件中显示图片的功能。

(4) 打开 AndroidManifest.xml 文件，添加以下代码：

```
…<uses-sdk
    android:minSdkVersion="8"
android:targetSdkVersion="18" />
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
  <application …
```

运行程序，效果如图 3-19 所示。



图 3-19 简易的图片浏览器

下面通过一个具体实例的实现过程来讲解使用基本布局控件的方法。

【例 3-14】 基本布局实例。

其实现步骤如下：

(1) 在 Eclipse 中新建一个 Android 应用项目，命名为 li3_14Layout。

(2) 编写布局文件，打开 res\layout 目录下的 main.xml 文件，将其代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <Button android:id = "@ + id/button0"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "使用 FrameLayout" />
    <Button android:id = "@ + id/button1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "使用 RelativeLayout" />
    <Button android:id = "@ + id/button2"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "使用 LinearLayout 和 RelativeLayout" />
    <Button android:id = "@ + id/button3"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "使用 TableLayout" />
</LinearLayout>
```

以上代码为一个典型的 LinearLayout 布局样式。

(3) 编写主文件，打开 src\fs. 3_14layout 包下的 MainActivity. java 文件，该文件主要用于调用公用文件实现具体的功能。其实现代码为：

```
public class MainActivity extends Activity
{
    OnClickListener listener0 = null;
    OnClickListener listener1 = null;
    OnClickListener listener2 = null;
    OnClickListener listener3 = null;
    Button button0;
    Button button1;
    Button button2;
    Button button3;
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        listener0 = new OnClickListener()
        {
            public void onClick(View v)
            {
```

```
                Intent intent0 = new Intent(MainActivity.this, ActivityFrameLayout.class);
                    setTitle("FrameLayout");
                    startActivity(intent0);
                }
            };
            listener1 = new OnClickListener()
            {
                public void onClick(View v)
                {
                Intent intent1 = new Intent(MainActivity.this, ActivityRelativeLayout.class);
                    startActivity(intent1);
                }
            };
            listener2 = new OnClickListener()
            {
                public void onClick(View v)
                {
                    setTitle("这是在 ActivityLayout");
                    Intent intent2 = new Intent(MainActivity.this, ActivityLayout.class);
                    startActivity(intent2);
                }
            };
            listener3 = new OnClickListener()
            {
                public void onClick(View v)
                {
                    setTitle("TableLayout");
                    Intent intent3 = new Intent(MainActivity.this, ActivityTableLayout.class);
                    startActivity(intent3);
                }
            };
            setContentView(R.layout.main);
            button0 = (Button) findViewById(R.id.button0);
            button0.setOnClickListener(listener0);
            button1 = (Button) findViewById(R.id.button1);
            button1.setOnClickListener(listener1);
            button2 = (Button) findViewById(R.id.button2);
            button2.setOnClickListener(listener2);
            button3 = (Button) findViewById(R.id.button3);
            button3.setOnClickListener(listener3);
        }
    }
```

在以上代码中，函数 setContentView(R.layout.main)用于实现 Activity 和 main.xml 的关联；button0、button1、button2、button3 代表了 4 个按钮，这 4 个按钮实现了引用，并给按钮设置了单击监听器，每一个监听器都跳转到一个新的 Activity。

（4）在 res\Layout 目录下新建一个名为 activityframelayout.xml 的文件，该文件实现第 1 个按钮 button0。单击 button0 按钮即会显示一个风景图，此界面为一个 FrameLayout 布局。其定义了这幅风景图的显示样式，即在 FrameLayout 布局中添加了一个图片显示控件 ImageView，实现代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<FrameLayout android:id="@+id/left"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ImageView android:id="@+id/photo" android:src="@drawable/bg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</FrameLayout>
```

(5) 在 res\Layout 目录下新建一个名为 relativelayout.xml 的文件,该文件用于实现单击第 2 个按钮 button1 后处理动作。单击 button1 按钮后会显示要求输入用户名的表单,此功能是通过 RelativeLayout 实现的。其代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Demonstrates using a relative layout to create a form -->
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:padding="10dip">
    <TextView android:id="@+id/label" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="请输入用户名: " />
    <!-- 这个 EditText 放置在上边 id 为 label 的 TextView 的下边 -->
    <EditText android:id="@+id/entry" android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label" />
    <!-- "取消"按钮和容器的右边齐平,并且设置左边的边距为 10dip -->
    <Button android:id="@+id/cancel" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip" android:text="取消" />
    <!-- "确定"按钮在"取消"按钮的左侧,并且和"取消"按钮的高度齐平 -->
    <Button android:id="@+id/ok" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/cancel"
        android:layout_alignTop="@id/cancel" android:text="确定" />
</RelativeLayout>
```

(6) 实现第 3 个按钮 button2 处理动作。在实例中,单击 button2 按钮即显示一系列文本,此功能是通过 LinearLayout 和 RelativeLayout 实现的。具体实现如下:

① 在 res\Layout 目录下新建一个名为 left.xml 的 RelativeLayout 布局文件,其实现第 a 组第 a 项\第 a 组第 b 项。其实现代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
android:id="@+id/left"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:id="@+id/view1"
        android:layout_width="fill_parent"
        android:layout_height="50px" android:text="第 a 组第 a 项" />
    <TextView android:id="@+id/view2"
```

```
        android:layout_width = "fill_parent"
        android:layout_height = "50px" android:layout_below = "@id/view1"
        android:text = "第 a 组第 b 项" />
</RelativeLayout>
```

② 在 res\layout 目录下新建一个名为 right.xml 的 RelativeLayout 布局文件，其实现第 b 组第 a 项和第 b 组第 b 项。其实现代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout android:id = "@ + id/right"
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <TextView android:id = "@ + id/right_view1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content" android:text = "第 b 组第 a 项" />
    <TextView android:id = "@ + id/right_view2"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_below = "@id/right_view1" android:text = "第 b 组第 b 项" />
</RelativeLayout>
```

③ 实现 Layout 与 Activity 的关联，即实现一个 Layout 和一个 Activity 的关联，而此 Layout 是在 XML 文件中定义的。在 Activity 中，为使用方便，可自行构建一个 Layout。根据需要在 res\fs.li3_14layout 包下新建 ActivityLayout.java 文件，代码为：

```
public class ActivityLayout extends Activity
{
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        LinearLayout layoutMain = new LinearLayout(this);
        layoutMain.setOrientation(LinearLayout.HORIZONTAL);
        setContentView(layoutMain);
        LayoutInflater inflate = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER
_SERVICE);
        RelativeLayout layoutLeft = (RelativeLayout) inflate.inflate(
                R.layout.left, null);
        RelativeLayout layoutRight = (RelativeLayout) inflate.inflate(
                R.layout.right, null);
        RelativeLayout.LayoutParams relParam = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
                RelativeLayout.LayoutParams.WRAP_CONTENT);
        layoutMain.addView(layoutLeft, 100, 100);
        layoutMain.addView(layoutRight, relParam);
    }
}
```

(7) 设计单击第 4 个按钮 button3 的处理动作。单击 button3 按钮会显示一个排列整齐的表单，此功能是通过 TableLayout 实现的。在 res\Layout 目录下新建一个 activitytablelayout.xml 文件，代码为：

```
<TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent" android:layout_height = "fill_parent"
    android:stretchColumns = "1">
    <TableRow>
        <TextView android:text = "用户名:" android:textStyle = "bold"
            android:gravity = "right" android:padding = "3dip" />
        <EditText android:id = "@ + id/username" android:padding = "3dip"
            android:scrollHorizontally = "true" />
    </TableRow>
    <TableRow>
        <TextView android:text = "密码:" android:textStyle = "bold"
            android:gravity = "right" android:padding = "3dip" />
        <EditText android:id = "@ + id/password" android:password = "true"
            android:padding = "3dip" android:scrollHorizontally = "true" />
    </TableRow>
    <TableRow android:gravity = "right">
        <Button android:id = "@ + id/cancel"
            android:text = "取消" />
        <Button android:id = "@ + id/login"
            android:text = "登录" />
    </TableRow>
</TableLayout>
```

（8）实现对应的布局，根据需要分别在 res\fs. li3_14layout 包下新建 ActivityRelativeLayout. java、ActivityTableLayout. java、ActivityFrameLayout. java 文件。

ActivityRelativeLayout. java 文件的代码为：

```
public class ActivityRelativeLayout extends Activity
{
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.relativelayout);
    }
}
```

ActivityTableLayout. java 文件的代码为：

```
public class ActivityTableLayout extends Activity
{
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);;
        setContentView(R.layout.activitytablelayout);
    }
}
```

ActivityFrameLayout. java 文件的代码为：

```
public class ActivityFrameLayout extends Activity
```

```
{
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);;
        setContentView(R.layout.activityframelayout);
    }
}
```

运行程序,效果如图 3-20 所示。



图 3-20 基本布局实例效果

第 4 章　Android 基本控件

无论看上去多么美观的 UI 界面，开始都是先创建容器，然后不断地向容器中添加界面控件，最后形成一个美观的 UI 界面。用户掌握这些基本控件是学好 Android 编程的基础。

4.1　文本类控件

Android 中提供了一些与文本输入相关的控件，这些控件不仅包括普通的文本框和编辑框，还包括为方便输入提供的自动完成文本框。

4.1.1　文本框属性及实例

1. 文本框属性

在 Android 中，文本框使用 TextView 表示，用于在屏幕上显示文本。这与 Java 中的文本框控件不同，它相当于 Java 中的标签，即 JLable。需要说明的是，Android 中的文本框控件可以显示单行文本、多行文本，以及带图像的文本。

在 Android 中，可以使用两种方法向屏幕中添加文本框，一种是通过在 XML 布局文件中使用<TextView>标记添加，另一种是在 Java 文件中通过 new 关键字创建。推荐使用第一种方法，即通过<TextView>标记在 XML 布局文件中添加。在 XML 布局文件中添加文本框的格式为：

```
< TextView
属性列表>
</TextView >
```

TextView 支持的常用 XML 属性如下。

- android:autoLink：设置当文本为 URL 链接、E-mail、电话号码、map 时，文本是否显示为可单击的链接。可选值有 none、web、email、phone、map、all。
- android:autoText：如果设置，将自动执行输入值的拼写纠正。此处无效果，在显示输入法并输入的时候起作用。
- android:bufferType：指定 getText()方法取得的文本类别。其选项 editable 类似于 StringBuilder，可追加字符，也就是说，getText 后可调用 append 方法设置文本内容，spannable 则可在给定的字符区域使用样式。
- android:capitalize：设置英文字母大写类型。此处无效果，需要显示输入法时才能看到，参见 EditView 中的此属性说明。

- android:cursorVisible：设定光标为显示/隐藏，默认显示。
- android:digits：设置允许输入哪些字符，例如 0～9、.、+、-、*、/、%、()。
- android:drawableBottom：在 text 的下方输出一个 drawable，例如图片。如果指定一个颜色，会把 text 的背景设为该颜色，并且和 background 同时使用时覆盖后者。
- android:drawableLeft：在 text 的左边输出一个 drawable，例如图片。
- android:drawablePadding：设置 text 和 drawable(图片)的间隔，与 drawableLeft、drawableRight、drawableTop、drawableBottom 一起使用，可设置为负数，单独使用没有效果。
- android:drawableRight：在 text 的右边输出一个 drawable。
- android:drawableTop：在 text 的正上方输出一个 drawable。
- android:editable：设置是否可编辑。
- android:editorExtras：设置文本的额外输入数据。
- android:ellipsize：设置当文字过长时该控件如何显示。其中，"start"表示省略号显示在开头；"end"表示省略号显示在结尾；"middle"表示省略号显示在中间；"marquee"表示以跑马灯的方式显示(动画横向移动)。
- android:freezesText：设置保存文本的内容以及光标的位置。
- android:gravity：设置文本位置，如设置成"center"，文本将居中显示。
- android:hintText：为空时显示的文字提示信息，可通过 textColorHint 设置提示信息的颜色。此属性在 EditView 中使用，但是这里也可以使用。
- android:imeOptions：附加功能，设置右下角 IME 动作与编辑框相关的动作，例如 actionDone 右下角将显示一个"完成"，而不设置默认是一个回车符号。
- android:imeActionId：设置 IME 动作 id。
- android:imeActionLabel：设置 IME 动作标签。
- android:includeFontPadding：设置文本是否包含顶部和底部的额外空白，默认为 true。
- android:inputMethod：为文本指定输入法，需要完全限定名(完整的包名)。
- android:inputType：设置文本的类型，用于帮助输入法显示合适的键盘类型。
- android:linksClickable：设置链接是否单击连接，即使设置了 autoLink。
- android:marqueeRepeatLimit：在 ellipsize 指定 marquee 的情况下，设置重复滚动的次数，当设置为 marquee_forever 时表示无限次。
- android:ems：设置 TextView 为 *N* 个字符的宽度。
- android:maxEms：设置 TextView 最长为 *N* 个字符的宽度。与 ems 同时使用时覆盖 ems 选项。
- android:minEms：设置 TextView 最短为 *N* 个字符的宽度。与 ems 同时使用时覆盖 ems 选项。
- android:maxLength：限制显示的文本长度，超出部分不显示。
- android:lines：设置文本的行数，设置两行就显示两行，即使第 2 行没有数据。
- android:maxLines：设置文本的最大显示行数，与 width 或者 layout_width 结合使用，超出部分自动换行，超出行数将不显示。

- android:minLines：设置文本的最小行数，与 lines 类似。
- android:lineSpacingExtra：设置行间距。
- android:lineSpacingMultiplier：设置行间距的倍数，例如“1.2”。
- android:numeric：如果被设置，该 TextView 有一个数字输入法。此处无用，设置后唯一的效果是 TextView 有单击效果。
- android:password：以小点“.”显示文本。
- android:phoneNumber：设置为电话号码的输入方式。
- android:privateImeOptions：设置输入法选项。
- android:scrollHorizontally：设置文本超出 TextView 的宽度的情况下是否出现横拉条。
- android:selectAllOnFocus：如果文本是可选择的，让它获取焦点而不是将光标移动到文本的开始位置或者末尾位置。在 TextView 中设置后无效果。
- android:shadowColor：指定文本阴影的颜色，需要和 shadowRadius 一起使用。
- android:shadow Dx：设置阴影横向坐标的开始位置。
- android:shadow Dy：设置阴影纵向坐标的开始位置。
- android:shadowRadius：设置阴影的半径。当设置为 0.1 时就变成字体的颜色了，一般设置为 3.0 的效果比较好。
- android:singleLine：设置单行显示。如果和 layout_width 一起使用，当文本不能全部显示时，后面用“…”来表示。例如：

```
android:text = "test_ singleLine"
android:singleLine = "true" android:layout_width = "20dp"
```

将只显示“t…”。如果不设置 singleLine 或者设置为 false，文本将自动换行。

- android:text：设置显示文本。
- android:textAppearance：设置文字外观。
- android:textColor：设置文本颜色。
- android:textColorHighlight：被选中文字的底色，默认为蓝色。
- android:textColorHint：设置提示信息文字的颜色，默认为灰色，和 hint 一起使用。
- android:textColorLink：文字链接的颜色。
- android:textScaleX：设置文字之间的间隔，默认为 1.0f。

android:textSize：设置文字大小，推荐度量单位“sp”，例如“15sp”。

- android:textStyle：设置字形（bold（粗体）：0、italic（斜体）：1、bolditalic（又粗又斜）：2），可以设置一个或多个，用“|”隔开。
- android:typeface：设置文本字体（normal：0、sans：1、serif：2、monospace（等宽字体）：3）。
- android:height：设置文本区域的高度，支持度量单位 px（像素）、dp、sp、in、mm（毫米）。
- android:maxHeight：设置文本区域的最大高度。
- android:minHeight：设置文本区域的最小高度。
- android:width：设置文本区域的宽度，支持度量单位 px（像素）、dp、sp、in、mm（毫米）。

- android:maxWidth：设置文本区域的最大宽度。
- android:minWidth：设置文本区域的最小宽度。

2. 文本框实例

下面给出一个关于文本框的实例。

【例 4-1】 利用 TextView 显示多种样式的文本。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li4_1TextView。

(2) 打开 res\layout 目录下的布局文件 main.xml，为 LinearLayout 设置背景，并为默认添加的 TextView 控件设置高度，对其中的 E-mail 格式的文本设置超链接。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/background02">
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "50px"
        android:text = "@string/hi"
        android:autoLink = "email"
        android:height = "50px" />
</LinearLayout >
```

(3) 在默认添加的 TextView 控件后面再添加一个 TextView 控件，该控件用于显示带图像的文本(图像在文字的上方)，代码为：

```
< TextView
    android:layout_width = "wrap_content"
    android:id = "@ + id/textView1"
    android:text = "带图片的 TextView"
    android:drawableTop = "@drawable/icon"
    android:layout_height = "wrap_content" />
```

(4) 在 TextView 控件后面再添加两个 TextView 控件，一个设置为可显示多行文本(默认)，另一个设置为只能显示单行文本，且将这两个 TextView 控件设置为不同颜色。代码为：

```
< TextView
    android:id = "@ + id/textView2"
    android:textColor = "＃09F"
    android:textSize = "20px"
    android:text = "多行文本：Android 是一种基于 Linux 的自由及开放源代码的操作系统，主要使
用于移动设备，由 Google 公司和开放手机联盟领导及开发。"
    android:width = "300px"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" />
< TextView
    android:id = "@ + id/textView3"
    android:textColor = "＃f00"
```

```
    android:textSize = "20px"
    android:text = "单行文本: Android 是一种基于 Linux 的自由及开放源代码的操作系统,主要使
用于移动设备,由 Google 公司和开放手机联盟领导及开发。"
    android:width = "300px"
    android:singleLine = "true"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" />
```

运行程序,效果如图 4-1 所示。



图 4-1　TextView 实例

4.1.2　编辑框属性及实例

1. 编辑框概述

在 Android 中,编辑框使用 EditText 表示,用于在屏幕上显示文本输入框,这与 Java 中的文本框控件功能类似。需要说明的是,Android 中的编辑框控件可以输入单行文本,也可以输入多行文本,还可以输入指定格式的文本(例如密码、电话号码、E-mail 地址等)。

在 Android 中可使用两种方法向屏幕中添加编辑框,一种是通过在 XML 布局文件中使用<EditText>标记添加。在 XML 布局文件中添加编辑框的格式为:

```
<EditText
 属性列表>
 </EditText>
```

由于 EditText 类是 TextView 的子类,所以对于 4.1.1 节的 TextViewXML 属性同样适用于 EditText 控件。特别要注意的是,在 EditText 控件中,android:inputType 属性可以帮助输入法显示合适的类型。例如,要添加一个密码框,可将 android:inputType 属性设置为 textPassword。

提示: *在 Eclipse 中打开布局文件,通过 Graphical Layout 视图,可以在可视化界面中*

添加编辑框控件，并在可视化界面中还列出了不同类型的输入框(例如密码框、数字密码框和输入电话号码的编辑框等)，只需要将其拖到布局文件中即可。

在屏幕中添加编辑框后，还需要获取编辑框中输入的内容，这可通过编辑框控件提供的getText()方法实现。在使用该方法时，要先获取到编辑框控件，然后再调用 getText()方法。例如，要获取布局文件中添加的 ID 属性为 login 的编辑框内容，可通过以下代码实现：

```
EditText login = (EditText)findViewById(R.id.login);
String loginText = login.getText().toString();
```

2. 编辑框实例

对于一个用户友好的输入界面而言，接受用户输入的文本框内默认会提示用户怎样输入：当用户把焦点切换到输入框时，输入框自动选中其中已输入的内容，避免用户删除已有内容；当用户把焦点切换到只接受电话号码的输入时，输入法自动切换到数字键盘。

【例 4-2】 实现用户的友好界面实例。

其实现步骤如下：

(1) 在 Eclipse 中创建一个名为 li4_2EditText 的 Android 应用项目。

(2) 打开 res\layout 目录下的 main.xml 布局文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1">
< TableRow >
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "用户名:"
    android:textSize = "10sp"
    android:background = "#000000"/>
< EditText
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:hint = "填写登录账号"
    android:selectAllOnFocus = "true"/>
</TableRow >
< TableRow >
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "密码:"
    android:textSize = "10pt"
    android:background = "#000000"/>
< EditText
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:password = "true"/>
```

```
</TableRow>
<TableRow>
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "电话号码:"
    android:textSize = "10pt"
    android:background = "#000000"/>
<EditText
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:hint = "填写您的电话号码"
    android:selectAllOnFocus = "true"
    android:phoneNumber = "true"/>
</TableRow>
<Button
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "注册"/>
</TableLayout>
```

以上布局文件中的第 1 个文本框通过 android:hint 指定了文本框的提示信息：填写登录账号——这是该文本框默认的提示。当用户还没有输入时，该文本框内默认显示这段信息。第 2 个文本框通过 android:password＝"true"设置为一个密码框，用户在该文本框中输入的字符以点号代替。第 3 个输入框通过 android:phoneNumber＝"true"设置为一个电话号码输入框。

运行程序，效果如图 4-2 所示。



图 4-2　登录界面

4.1.3 自动文本框属性及实例

1. 自动文本框的属性

自动文本框使用 AutoCompleteTextView 表示，实现用户输入一定字符后显示一个下拉菜单，供用户从中选择，当用户选择某个菜单项后，按用户所选自动填写该文本框。

在屏幕中添加自动文本框，可在 XML 布局文件中通过<AutoCompleteTextView>标记添加，格式为：

```
<AutoCompleteTextView
 属性列表>
</AutoCompleteTextView>
```

AutoCompleteTextView 控件继承自 TextView，所以它支持 EditText 控件提供的属性，同时，该控件还支持以下 XML 属性。

- android:completionHint：用于为弹出的下拉菜单指定提示标题。
- android:completionThreshold：用于指定用户至少输入几个字符才会显示提示。
- android:dropDownHeight：用于指定下拉菜单的高度。
- android:dropDownHorizontalOffset：用于指定下拉菜单与文本之间的水平偏移。下拉菜单默认与文本框左对齐。
- android:dropDownVerticalOffset：用于指定下拉菜单与文本之间的垂直偏移。下拉菜单默认与文本框左对齐。
- android:dropDownWidth：用于指定下拉菜单的宽度。
- android:popupBackground：用于为下拉菜单设置背景。

2. 自动文本框实例

下面通过一个例子来说明自动文本框的使用。

【例 4-3】 利用自动文本框实现歌曲的选择。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li4_3Auto。

(2) 打开 res\layout 目录下的 main.xml 布局文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "@drawable/bj1">
    <AutoCompleteTextView
        android:id = "@ + id/autoCompleteTextView1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:completionHint = "请选择你喜欢的歌曲"
        android:completionThreshold = "1"
        android:dropDownHorizontalOffset = "20dp"
        android:ems = "10"
        android:text = "AutoCompleteTextView" >
```

```
            <requestFocus />
        </AutoCompleteTextView>
    </LinearLayout>
```


(3) 打开 src\fs.li4_3auto 目录下的 MainActivity.java 文件，首先获取布局文件中的自动文本框，然后创建一个保存下拉菜单中要显示的菜单项的 ArrayAdapter 适配器，最后将该适配器与自动文本相关联。代码为：

```
public class MainActivity extends Activity {
    // 定义字符串数组作为提示的文本
    String[] books = new String[] { "Roar", "Roarholt", "Rain", "Raining" };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // 创建一个 ArrayAdapter 封装数组
        ArrayAdapter<String> av = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line, books);
        AutoCompleteTextView auto = (AutoCompleteTextView)
        findViewById(R.id.autoCompleteTextView1);
        auto.setAdapter(av);
    }
}
```

运行程序，在屏幕上会显示由自动文本框提供的搜索框，在自动文本框中输入相应的首字母，即可弹出对应的选择，效果如图 4-3 所示。



图 4-3　自动文本框实例

4.2 按钮类控件

在 Android 中提供了一些按钮类控件，主要包括普通按钮、图片按钮、单选按钮和多选按钮，下面进行介绍。

4.2.1 普通按钮概述及实例

1. 普通按钮概述

普通按钮在 Andriod 中用 Button 表示。Button 控件继承自 TextView 类，用户可以对 Button 控件进行按下或单击等操作，Button 控件的用法比较简单，主要是为 Button 控件设置 View. OnClickListener 监听器并在监听的实现代码中开发按钮按下事件的处理代码。

Button 控件除了可以在按钮上显示字符串外，还可以通过修改背景来显示图片等 Drawable 资源。

在 Android 中可以使用两种方法向屏幕中添加按钮，一种是通过在 XML 布局文件中使用<Button>标记添加，另一种是在 Java 文件中通过 new 关键字创建。推荐使用第一种方法，也就是通<Button>标记在 XML 布局文件中添加。在 XML 布局文件中添加普通按钮的基本格式为：

```
<Button
     android:id = "@ + id/button1"
     android:layout_width = "wrap_content"
     android:layout_height = "wrap_content"
     android:layout_alignTop = "@ + id/textView1"
     android:layout_marginLeft = "46dp"
     android:layout_toRightOf = "@ + id/textView1"
     android:text = "Button" />
```

在屏幕上添加按钮后，还需要为按钮添加单击事件监听器，这样才能让按钮发挥其特有的用途。在 Android 中提供了两种为按钮添加单击事件监听器的方法，一种是在 Java 代码中完成。例如，在 Activity 的 onCreate()方法中完成，代码为：

```
import android.view.View.OnClickListener;
import android.widget.Button;
Button login = (Button)findViewByld(R.id.login);      //通过 id 获取布局文件中添加的按钮
login.setOnClickListener(new onClickListener(){       //为按钮添加单击事件监听器
    @Override
    public void onClick(View v){
        //编写要执行的动作代码
        }
});
```

另一种是在 Activity 中编写包含 View 类型参数的方法，并且将要触发的动作代码放在该方法中，然后在布局文件中通过 android:onClick 属性指定对应的方法名实现。例如，在 Activity 中编写一个 butClick()方法，关键代码为：

```
public void butClick(View view)
```

```
{
    //编写要执行的动作代码
}
```


那么,即可在布局文件中通过 android:onClick="butClick"为按钮添加单击事件监听器。

2. 普通按钮实例

下面通过一个实例来介绍怎样添加普通按钮,并通过两种方法为按钮添加单击事件监听器。

【例 4-4】 为按钮添加事件监听器实例。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 li4_4Button。

(2) 单击 res\layout 目录下的布局文件 main. xml,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "horizontal"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/bj1" >
    < Button android:text = "登录"
        android:id = "@ + id/login"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
<! -- 设置 android:onClick 属性,指定一个单击事件监听器 -->
    < Button
        android:id = "@ + id/register"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:onClick = "butClick"
        android:text = "注册" />
</LinearLayout >
```

(3) 打开 src\fs. li4_4button 目录下的 MainActivity. java 文件,代码为:

```
public class MainActivity extends Activity {
    @Override
    //为 login 按钮添加单击事件监听器
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //通过 ID 获取布局文件中添加的按钮
        Button login = (Button)findViewById(R.id.login);
        login.setOnClickListener(new OnClickListener() {//为按钮添加单击事件监听
            //用于指定将要触发的动作
            @Override
            public void onClick(View v) {
                Toast toast = Toast.makeText(MainActivity.this, "单击了"登录"按钮", Toast.
LENGTH_SHORT);
                toast.show();

            }
        });
```

```
    }
    public void butClick(View view){
        Toast toast = Toast.makeText(MainActivity.this, "单击了"注册"按钮", Toast.LENGTH_
SHORT);
        toast.show();
    }
}
```

运行程序,效果如图 4-4 所示。单击“登录”按钮,将显示“单击了“登录”按钮”的提示信息;单击“注册”按钮,将显示“单击了“注册”按钮”的提示信息。



图 4-4　为按钮添加单击事件监听器

4.2.2　图片按钮概述与实例

1. 图片按钮概述

ImageButton 控件继承自 ImageView 类,ImageButton 控件与 Button 控件的主要区别是在 ImageButton 中没有 text 属性,即按钮将显示图片而不是文本。在 ImageButton 中设置按钮显示的图片可以通过 adnroid:src 属性来设置,也可以通过 setImageResource(int)方法来设置。

默认情况下,ImageButton 和 Button 一样具有背景色,当按钮处于不同的状态(如按下等)时,背景色也会随之变化。当 ImageButton 所显示的图片不能完全覆盖背景色时,这种显示效果将会非常糟糕,所以使用 ImageButton 时一般将背景色设置为其他图片或直接设置为透明的。

不管怎样,都需要为按钮控件指定不同状态下显示的图片,否则用户将无法区别是否按下了按钮。设置按钮在不同状态下显示不同的图片可以通过编写 XML 文件来实现。

2. 图片按钮实例

下面通过一个例子来说明怎样为 ImageButton 控件的不同状态设置不同的显示图片。

【例 4-5】 图片按钮的演示。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li4_5ImageBut。

(2) 打开 res\layout 目录下的 main. xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1"
    android:gravity = "center_horizontal"
    android:orientation = "vertical" >
    <! -- 按钮的图片为 a1 -- >
    < ImageButton
        android:id = "@ + id/imageButton1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:src = "@drawable/b" />
    <! -- 按钮的图片为 b1 -- >
    < ImageButton
        android:id = "@ + id/imageButton2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:background = "#0FFF"
        android:onClick = "imageClick"
        android:src = "@drawable/b1" />
</LinearLayout >
```

(3) 打开 src\fs. li4_5imagebut 目录下的 MainActivity. java 文件，为第一个图片按钮设置单击事件监听器，并编写方法 imageClick()，用于指定将要触发的动作代码。具体代码为：

```
public class MainActivity extends Activity {
    @Override
    //为 login 按钮添加单击事件监听
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button login = (Button)findViewById(R.id.login);
        //通过 ID 获取布局文件中添加的按钮
       login.setOnClickListener(new OnClickListener() {//为按钮添加单击事件监听
            //用于指定将要触发的动作
            @Override
            public void onClick(View v) {
                Toast toast = Toast.makeText(MainActivity.this, "单击了"登录"按钮", Toast.
LENGTH_SHORT);
                toast.show();
            }
        });
    }
    public void butClick(View view){
```

```
            Toast toast = Toast.makeText(MainActivity.this, "单击了"注册"按钮", Toast.LENGTH_
SHORT);
            toast.show();
        }
    }
```

运行程序,效果如图 4-5 所示。当单击了某一个按钮时,界面中会显示对应的信息。



图 4-5　图片按钮实例

4.2.3　开关按钮属性及实例

1. 开关按钮属性

ToogleButton 的继承关系如图 4-6 所示。ToogleButton 的状态只能是选中和未选中状态,并且需要为不同的状态设置不同的显示文本。除了继承自父类的一些属性和方法外,ToggleButton 还具有一些自己的 XML 属性,如下所述。

- android:checked:设置该按钮是否被选中。
- android:textOff:设置当该按钮没有被选中时显示的文本。
- android:textOn:设置当该按钮被选中时显示的文本。

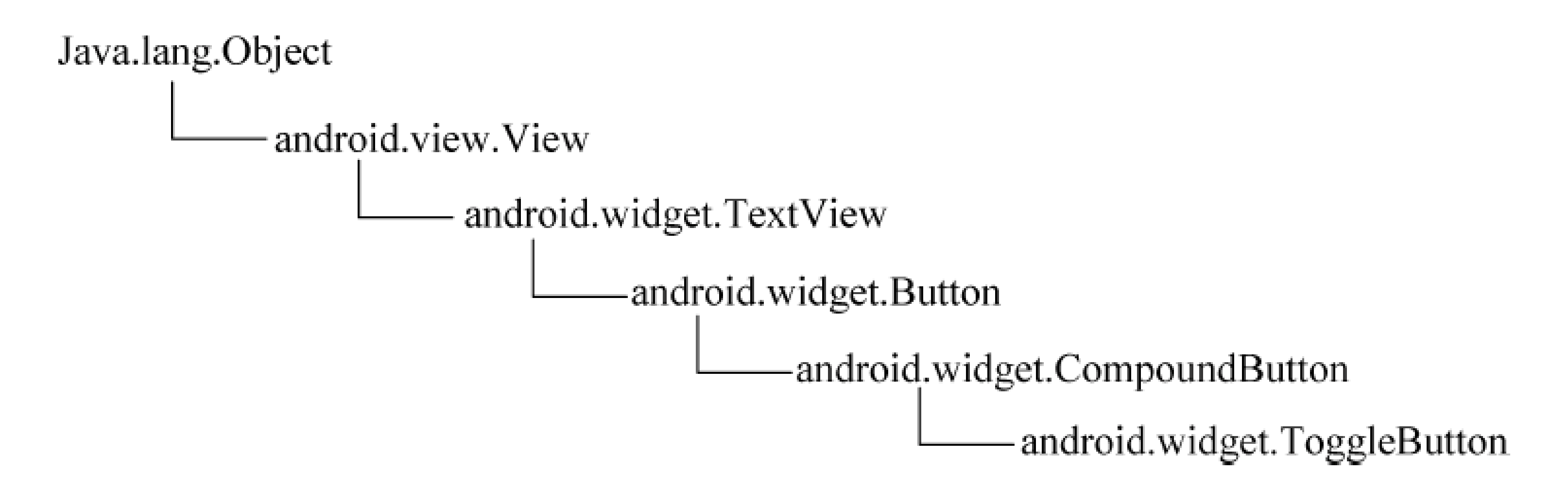


图 4-6　开关按钮的继承关系

2. 开关按钮实例

下面通过一个实例来说明开关按钮的使用。

【例 4-6】 用 ToggleButton 控制灯泡的开与关。

其实现步骤如下：

(1) 在 Eclipse 中创建一个名为 li4_6Toggle 的 Android 应用项目。

(2) 打开 res\layout 目录下的 main.xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
 <! -- 声明一个垂直分布的线性分布 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = "@drawable/bj1">
<! -- 声明一个 ImageView 控件,该控制将会根据 ToggleButton 的状态显示不同的图片 -->
< ImageView
    android:id = "@ + id/imageView"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:src = "@drawable/bulb_off"
    android:layout_gravity = "center_horizontal"/>
 <! -- 声明一个 ToggleButton 控件,设置该控件在选中和未选中状态下显示的字符串 -->
< ToggleButton
    android:id = "@ + id/toggleButton"
    android:layout_width = "140dip"
    android:layout_height = "wrap_content"
     android:textOn = "开灯"
     android:textOff = "关灯"
     android:layout_gravity = "center_horizontal"/>
</LinearLayout >
```

(3) 打开 src\fs. li4_6toggle 目录下的 MainActivity. java，代码为：

```
public class MainActivity extends Activity {
    private ImageView imageView = null;
    private ToggleButton toggleButton = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //设置图铃状态
        imageView = (ImageView) findViewById(R.id.imageView);
        //设置 ToggleButton 状态
        toggleButton = (ToggleButton)findViewById(R.id.toggleButton);
        toggleButton.setOnCheckedChangeListener(new OnCheckedChangeListener(){
            //重写 onCheckedChanged 方法
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked){
                            toggleButton.setChecked(isChecked); //控制控件状态
                //设置图像资源
imageView.setImageResource(isChecked?R.drawable.bulb_off:R.drawable.bulb_on);
            }
        });
    }
}
```

运行程序，效果如图 4-7 所示，单击图中的“开灯”按钮，效果如图 4-8 所示。



图 4-7　默认状态



图 4-8　开灯效果

4.2.4　单选按钮/复选框属性及实例

1. 单选按钮概述

在默认情况下，单选按钮显示为一个圆形图标，并且在该图标旁边放置一些说明性文字。而在程序中，一般将多个单选按钮放置在按钮组中，使这些单选按钮表现出某种功能，当用户选中某个单选按钮后，按钮组中的其他按钮将被自动取消选中状态。在 Android 中，单选按钮使用 RadioButton 表示，而 RadioButton 类又是 Button 的子类，所以单选按钮可以直接使用 Button 支持的各种属性。

在 Android 中可以使用两种方法向屏幕中添加单选按钮，一种是通过在 XML 布局文件中使用<RadioButton>标记添加，另一种是在 Java 文件中通过 new 关键字创建。推荐使用第一种方法，也就是通过<RadioButton>在 XML 布局文件中添加。在 XML 布局文件中添加单选按钮的格式为：

```
<RadioButton
    android:id="@+id/radioButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/textView1"
    android:layout_below="@+id/textView1"
    android:layout_marginLeft="23dp"
    android:layout_marginTop="95dp"
    android:text="RadioButton"
    android:checked="true|false" />
```

RadioButton 控件的 android:checked 属性用于指定选中状态，当属性值为 true 时，表

示选中；当属性值为 false 时，表示不选中，默认为 false。

通常情况下，RadioButton 控件需要和 RadioGroup 控件一起使用，组成一个单选按钮组。在 XML 布局文件中，添加 RadioGroup 控件的格式为：

```
<RadioGroup
    android:id = "@id/radioGroup1"
    android:orientation = "horizontal"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content">
  <! -- 添加多个 RadioButton 控件 -->
```

2. 复选框概述

在默认情况下，复选框显示为一个方块图标，并且在该图标旁边放置一些说明性文字。与单选按钮唯一不同的是，复选框可以进行多选设置，每一个复选框都提供"选中"和"不选中"两种状态。在 Android 中，复选框使用 CheckBox 表示，而 CheckBox 类又是 Button 的子类，所以复选框可以直接使用 Button 支持的各种属性。

在 Android 中可以使用两种方法向屏幕中添加复选框，一种是通过在 XML 布局文件中使用<CheckBox>标记添加，另一种是在 Java 文件中通过 new 关键字创建。推荐使用第一种方法，即通过<CheckBox>在 XML 布局文件中添加。在 XML 布局文件中添加复选框的格式为：

```
<CheckBox
    android:id = "@ + id/checkBox1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignParentLeft = "true"
    android:layout_alignParentTop = "true"
    android:layout_marginLeft = "56dp"
    android:layout_marginTop = "98dp"
    android:text = "CheckBox" />
```

由于复选框可以选中多项，所以为了确定用户是否选择了某一项，还需要为每一个选项添加事件监听器。例如，要为 id 为 ch1 的复选框添加状态改变事件监听器，实现代码为：

```
final CheckBox ch1 = (CheckBox)findViewByld(R.id.ch1);  //根据 id 属性获取复选框
ch1.setOnCheckedChangeListener(new OnCheckedChangeListener(){
    @Override
    public void onCheckedChanged(CompoundButton buttonView,boolean isChecked){
        if(ch1.isChecked()){                         //判断该复选框是否被选中
            ch1.getText();                           //获取选中项的值
            }
      }
})
```

3. 单选按钮与复选框实例

下面通过一个例子来演示单选按钮及复选框的用法。

【例 4-7】 利用单选按钮和复选框控制灯光的开与关。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_7RadioCheck。

(2) 打开 res\layout 目录下的 main.xml 文件,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< ScrollView xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:scrollbars = "vertical">
    < LinearLayout android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
        <! -- RadioButton 控件演示 -- >
        < ImageView android:id = "@ + id/imageView01"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/bulb_on"
            android:layout_gravity = "center_horizontal" />
            < RadioGroup android:id = "@ + id/radioGroup"
                android:orientation = "horizontal"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:layout_gravity = "center_horizontal">
                < RadioButton android:id = "@ + id/on"
                    android:text = "开灯"
                    android:layout_width = "wrap_content"
                    android:layout_height = "wrap_content"
                    android:checked = "true" />
                    < RadioButton android:id = "@ + id/off"
                        android:text = "关灯"
                        android:layout_width = "wrap_content"
                        android:layout_height = "wrap_content" />
                        </RadioGroup >
                    <! -- CheckBox 控件演示 -- >
                     < ImageView android:id = "@ + id/imageView02"
                        android:layout_width = "wrap_content"
                        android:layout_height = "wrap_content"
                        android:src = "@drawable/bulb_on"
                        android:layout_gravity = "center_horizontal" />
                        < CheckBox android:id = "@ + id/checkBox"
                            android:text = "开灯"
                            android:checked = "true"
                            android:layout_width = "wrap_content"
                            android:layout_height = "wrap_content"
                            android:layout_gravity = "center_horizontal" />
            </LinearLayout >
    </ScrollView >
```

(3) 打开 src\fs.li4_7radiocheck 目录下的 MainActivity.java 文件,代码为:

```
public class MainActivity extends Activity {
    private ImageView imageView01 = null;
    private ImageView imageView02 = null;
    private CheckBox checkBox = null;
    private RadioButton on = null;                    //开灯
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imageView01 = (ImageView)findViewById(R.id.imageView01);
        imageView02 = (ImageView)findViewById(R.id.imageView02);
        checkBox = (CheckBox)findViewById(R.id.checkBox);
        on = (RadioButton)findViewById(R.id.on);
        on.setOnCheckedChangeListener(listener);
        checkBox.setOnCheckedChangeListener(listener);
        }
        OnCheckedChangeListener listener = new OnCheckedChangeListener(){
            public void onCheckedChanged(CompoundButton buttonView,boolean isChecked){
                if(buttonView instanceof RadioButton){

imageView01.setImageResource(isChecked?R.drawable.bulb_on:R.drawable.bulb_off);
                    }else if(buttonView instanceof CheckBox){
                        checkBox.setText(isChecked?"开灯":"关灯");
imageView02.setImageResource(isChecked?R.drawable.bulb_on:R.drawable.bulb_off);
                        }
                    }
            };
    }
```

运行程序，效果如图 4-9 所示。当选择图中的“关灯”单选按钮，并去掉复选框前面的“√”符号时，效果如图 4-10 所示。



图 4-9 开灯状态



图 4-10 关灯状态

4.3 列表类控件

在 Android 中提供了两种列表类控件，一种是列表选择框，通常用于实现类似于网页中常见的下拉列表框；另一种是列表视图，通常用于实现在一个窗口中只显示一个列表。

4.3.1 列表选择框属性及实例

1. 列表选择框属性

Android 中提供的列表选择框(Spinner)相当于在网页中常见的下拉列表框,通常用于提供一系列可选择的列表项,供用户选择,从而方便用户。

在 Android 中可以使用两种方法向屏幕中添加列表选择框,一种是通过在 XML 布局文件中使用<Spinner>添加。在 XML 布局文件中添加列表选择框的格式为:

```
<Spinner
    android:prompt = "@string/info"
    android:id = "@ + id/ID 号 "
    android:entries = "@array/数组名称"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content" />
```

其中,android:entries 为可选属性,用于指定列表项,如果不在布局文件中指定该属性,可以在 Java 代码中通过为其指定适配器的方式指定;android:prompt 属性也是可选属性,用于指定列表选择框的标题。

通常情况下,如果列表选择框中要显示的列表项是可知的,那么将其保存在数组资源文件中,然后通过数组资源为列表选择框指定列表项,这样,就可以在不编写 Java 代码的情况下实现一个列表选择框。

2. 列表选择框实例

下面通过一个实例来演示列表选择框的使用。

【例 4-8】 演示列表选择框的使用。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_8spinner。

(2) 打开 res\layout 目录下的 main. xml 文件,声明一个 TextView 控件和一个 Spinner 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
 xmlns:android = "http://schemas.android.com/apk/res/android"
 android:layout_width = "fill_parent"
 android:layout_height = "fill_parent"
 android:orientation = "vertical"
 android:background = "@drawable/bj1">
 <TextView android:id = "@ + id/spinnerText"
           android:layout_width = "fill_parent"
           android:layout_height = "wrap_content"></TextView>
 <Spinner android:id = "@ + id/Spinner01"
          android:layout_width = "fill_parent"
          android:layout_height = "wrap_content"></Spinner>
</LinearLayout>
```

(3) 打开 src\fs. li4_8spinner 目录下的 MainActivity. java 文件,通过列表框实现血型的选择。代码为:

```
public class MainActivity extends Activity {
```

```
    private static final String[] m = {"A 型","B 型","O 型","AB 型","其他"};
    private TextView view ;
    private Spinner spinner;
    private ArrayAdapter<String> adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO 自动生成的方法存根
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        view = (TextView) findViewById(R.id.spinnerText);
        spinner = (Spinner) findViewById(R.id.Spinner01);
        //将可选内容与 ArrayAdapter 连接起来
        adapter = new ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,m);
        //设置下拉列表的风格
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        //将 Adapter 添加到 Spinner 中
        spinner.setAdapter(adapter);
        //添加 Spinner 事件监听
        spinner.setOnItemSelectedListener(new SpinnerSelectedListener());
        //设置默认值
        spinner.setVisibility(View.VISIBLE);
    }
    //使用数组形式操作
    class SpinnerSelectedListener implements OnItemSelectedListener{
        public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2,
                long arg3) {
            view.setText("你的血型是: " + m[arg2]);
        }
        public void onNothingSelected(AdapterView<?> arg0) {
        }
    }
}
```

运行程序,效果如图 4-11 所示。当单击 Spinner 左侧的三角符号时,将弹出选择列表,效果如图 4-12 所示。




图 4-11　默认状态效果图



图 4-12　行列列表框

4.3.2 列表视图属性及实例

1. 列表视图属性

列表视图是 Android 中最常用的一种视图控件,它以垂直列表的形式列出需要显示的列表项。例如,显示系统设置项或功能内容列表等。在 Android 中可以使用两种方法向屏幕中添加列表视图,一种是直接使用 ListView 控件创建,另一种是让 Activity 继承 ListActivity 实现。下面分别介绍。

1) 直接使用 ListView 控件创建

直接使用 ListView 控件创建列表视图也有两种方式,一种是通过在 XML 布局文件中使用<ListView>标记添加,另一种是在 Java 文件中通过 new 关键字创建。推荐使用第一种方法,即通过<ListView>在 XML 布局文件中添加。在 XML 布局文件中添加 ListView 的格式为:

```
<ListView
  属性列表>
</ListView>
```

ListView 控件支持的常用 XML 属性如下:

- android:divider: 用于为列表视图设置分隔条,既可以用颜色分隔,也可以用 Drawable 资源分隔。
- android:dividerHeight: 用于设置分隔条的高度。
- android:entries: 用于通过数组资源为 ListView 指定列表项。
- android:footerDividersEnabled: 用于设置是否在 footer View 之前绘制分隔条,默认值为 true,当设置为 false 时表示不绘制。在使用该属性时,需要通过 ListView 控件提供的 addPooterView()方法为 ListView 设置 footer View。
- android:headerDividersEnabled: 用于设置是否在 header View 之后绘制分隔条,默认值为 true,当设置为 false 时表示不绘制。在使用该属性时,需要通过 ListView 控件提供的 addHeaderView()方法为 ListView 设置 header View。

2) 让 Activity 继承 ListActivity 实现

如果程序的窗口中仅需要显示一个列表,则可以直接让 Activity 继承 ListActivity 来实现。在继承了 ListActivity 的类中无须调用 setContentView()方法显示页面,而是直接为其设置适配器,从而显示一个列表。

2. 列表视图实例

下面通过一个实例来演示列表视图的实例。

【例 4-9】 实现一个 ListView,ListView 里面有标题、内容和图片,并加入单击和长按响应。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_9ListView。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中定义一个 ListView。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
```

```
        android:id = "@ + id/LinearLayout01"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        xmlns:android = "http://schemas.android.com/apk/res/android"
        android:background = "@drawable/bj1">
    < ListView android:layout_width = "wrap_content"
               android:layout_height = "wrap_content"
               android:id = "@ + id/ListView01" />
    </LinearLayout >
```

(3) 在 res\layout 目录下新建一个 list_items. xml 布局文件,对于 ListView 每个项目的 Layout 用 RelativeLayout 实现。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    < ImageView
    android:paddingTop = "12dip"
    android:layout_alignParentRight = "true"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:id = "@ + id/ItemImage"/>
< TextView
    android:text = "TextView01"
    android:layout_height = "wrap_content"
    android:textSize = "20dip"
    android:layout_width = "fill_parent"
    android:id = "@ + id/ItemTitle"/>
< TextView
    android:text = "TextView02"
    android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:layout_below = "@ + id/ItemTitle"
    android:id = "@ + id/ItemText" />
</RelativeLayout >
```

(4) 打开 src\fs. li4_9listview 目录下的 MainActyivity. java 文件,在 Activity 里面调用和加入 Listener。代码为:

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //绑定 Layout 里面的 ListView
        ListView list = (ListView) findViewById(R.id.ListView01);
        //生成动态数组,加入数据
```

```
        ArrayList < HashMap < String, Object >> listItem  =  new ArrayList < HashMap < String,
Object >>();
        for(int i = 0;i < 10;i++)
        {
            HashMap < String, Object > map  =  new HashMap < String, Object >();
            map.put("ItemImage", R.drawable.ic_launcher);                  //图像资源的 id
            map.put("ItemTitle", "Level " + i);
            map.put("ItemText", "Finished in 1 Min 54 Secs, 70 Moves! ");
            listItem.add(map);
        }
        //生成适配器的 Item 和动态数组对应的元素
        SimpleAdapter listItemAdapter  =  new SimpleAdapter(this,listItem,    //数据源
            R.layout.list_items,//ListItem 的 XML 实现
            //动态数组与 ImageItem 对应的子项
            new String[] {"ItemImage","ItemTitle", "ItemText"},
            //ImageItem 的 XML 文件里面有一个 ImageView、两个 TextView id
            new int[] {R.id.ItemImage,R.id.ItemTitle,R.id.ItemText}
        );
        //添加并且显示
        list.setAdapter(listItemAdapter);
        //添加单击
        list.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                    long arg3) {
                setTitle("单击第" + arg2 + "个项目");
            }
        });
      //添加长按单击
        list.setOnCreateContextMenuListener(new OnCreateContextMenuListener() {
            @Override
           public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo) {
                menu.setHeaderTitle("长按菜单 - ContextMenu");
                menu.add(0, 0, 0, "弹出长按菜单 0");
                menu.add(0, 1, 0, "弹出长按菜单 1");
            }
        });
    }
    //长按菜单响应函数
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        setTitle("单击了长按菜单里面的第" + item.getItemId() + "个项目");
        return super.onContextItemSelected(item);
    }
}
```

运行程序,默认效果如图 4-13 所示,长按鼠标将弹出菜单,如图 4-14 所示。




图 4-13　默认效果



图 4-14　长按鼠标效果

4.4　图像类控件

在 Android 中提供了比较丰富的图像类控件，例如用来显示图片的图像视图、用来浏览图片的网格视图，以及图像切换器和画廊视图等。

4.4.1　图像视图属性及实例

1. 图像视图属性

图像视图使用 ImageView 表示，用于在屏幕中显示任何的 Drawable 对象，通常用来显示图片。在 Android 中可以使用两种方法向屏幕中添加图像视图，一种是通过在 XML 布局文件中使用<ImageView>标记添加，另一种是在 Java 文件中通过 new 关键字创建。推荐使用第一种方法。

在使用 ImageView 控件显示图像时，通常将要显示的图片放置在 res\drawable 目录中，然后用以下代码将其显示在布局管理器中。

```
< ImageView
  属性列表>
</ ImageView >
```

ImageView 控件支持的常用 XML 属性如下。

- android:adjustViewBounds：设置是否保持宽高比，需要和 maxWidth、MaxHeight 一起使用，单独使用没有效果。
- android:cropToPadding：设置是否截取指定区域用空白代替，单独设置无效果，需要和 scrollY 一起使用。

• android:maxHeight：设置 ImageView 的最大高度，需要设置 android:adjustViewBounds 属性的值为 true，否则不起作用。
• android:maxWidth：设置 ImageView 的最大宽度，需要设置 android:adjustViewBounds 属性的值为 true，否则不起作用。
• android:scaleType：用于设置所显示的图片怎样缩放和移动以适应 ImageView 的大小，其属性值可以是 maxtrix（使用 matrix 方式进行缩放）、fitXY（对图片横向、纵向独立缩放，使得该图片完全适应于该 ImageView，图片的纵横比可能会改变）、fitStart（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的左上角）、fitCenter（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的中央）、fitEnd（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的右下角）、center（把图像放在 ImageView 的中间，但不进行任何缩放）、centerCrop（保持纵横比缩放图片，使得图片能完全覆盖 ImageView）或 centerInside（保持纵横比缩放图片，使得 ImageView 能完全显示该图片）。
• android:src：设置 View 的 drawable（例如图片，也可以是颜色，但是需要指定 View 的大小）。
• android:tint：用于为图片着色，其属性值可以是＃rgb、＃argb、＃rrggbb、＃aarrggbb 表示的颜色值。

2. 图像视图实例

下面通过一个实例来说明图像视图的用法。

【例 4-10】 利用 ImageView 实现手机模拟图片查看器。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_10ImageView。

(2) 打开 res\layout 目录下的 main. xml 布局文件，添加 4 个按钮控件和一个 ImageView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = "@drawable/bj1">
< ImageView android:id = "@ + id/imageView"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:layout_gravity = "center_horizontal"
android:src = "@drawable/b"/>
< LinearLayout
android:orientation = "horizontal"
android:layout_width = "fill_parent"
android:layout_height = "wrap_content"
android:layout_gravity = "center_horizontal">
< Button
    android:id = "@ + id/previous"
    android:layout_width = "match_parent"
```

```
        android:layout_height = "wrap_content"
        android:layout_gravity = "end"
        android:text = "上一张" />
</LinearLayout>
<Button
        android:id = "@ + id/alpha_minus"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "减少透明度" />
<Button
        android:id = "@ + id/next"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "下一张" />
<Button
        android:id = "@ + id/alpha_plus"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "增加透明度" />
</LinearLayout>
```

(3) 打开 src\fs. li4_10imageview 目录下的 MainActivity. java 文件，为 ImageView 实现事件监听器。实现代码为：

```
public class MainActivity extends Activity {
    private ImageView imageView = null;
    private Button previous = null;                //上一张
    private Button next = null;                    //下一张
    private Button alpha_plus = null;              //增加透明度
    private Button alpha_minus = null;             //减少透明度
    private int currentImgId = 0;                  //记录当前 ImageView 显示的图片的 ID
    private int alpha = 255;                       //记录 ImageView 的透明度
    int [] imgId = {
        //ImageView 显示的图片数组
        R.drawable.b1,
        R.drawable.b2,
        R.drawable.b3,
        R.drawable.b4,
        R.drawable.b5,
        };
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            imageView = (ImageView)findViewById(R.id.imageView);
            previous = (Button)findViewById(R.id.previous);
            next = (Button)findViewById(R.id.next);
            alpha_plus = (Button)findViewById(R.id.alpha_plus);
            alpha_minus = (Button)findViewById(R.id.alpha_minus);
            previous.setOnClickListener(listener);
            next.setOnClickListener(listener);
            alpha_plus.setOnClickListener(listener);
            alpha_minus.setOnClickListener(listener);
        }
```

```
        private View.OnClickListener listener = new View.OnClickListener(){
            public void onClick(View v) {
                if(v == previous){
                    currentImgId = (currentImgId - 1 + imgId.length) % imgId.length;
                    imageView.setImageResource(imgId[currentImgId]);
                    }
                    if(v == next){
                        currentImgId = (currentImgId + 1) % imgId.length;
                        imageView.setImageResource(imgId[currentImgId]);
                    }
                    if(v == alpha_plus){
                        alpha += 10;
                        if(alpha > 255){
                            alpha = 255;
                        }
                        imageView.setAlpha(alpha);
                    }
                    if(v == alpha_minus){
                        alpha -= 10;
                        if(alpha < 0){
                            alpha = 0;
                    }
                    imageView.setAlpha(alpha);
                }
            }
        };
    }
```

运行程序,效果如图 4-15 所示。当单击“增加透明度”按钮时图片颜色逐渐变深,当单击“减小透明度”按钮时图片颜色逐渐变淡。



图 4-15　手机图片浏览器

4.4.2 网格视图属性及实例

1. 网格视图属性

网格视图按照行、列布局方式来显示多个控件，通常用于显示图片或图标等。在使用网格视图时，首先需要在屏幕上添加 GridView 控件，通常使用<GridView>标记在 XML 布局文件中添加。在 XML 布局文件中添加网格视图的格式为：

```
<GridView
  属性列表>
</GridView>
```

GridView 控件支持的 XML 属性如下。

- android:columnWidth：用于设置列的宽度。
- android:gravity：用于设置对齐方式。
- android:horizontalSpacing：用于设置各元素之间的水平间距。
- android:numColumns：用于设置列数，其属性值通常为大于 0 的值，如果只有一列，那么最好使用 ListView 实现。
- android:stretchMode：用于设置拉伸模式，其属性值可以是 none（不拉伸）、spacingWidth（仅拉伸元素之间的间距）、columnWidth（仅拉伸表格元素本身）或 spacingWidthUniform（表格元素本身、元素之间的间距一起拉伸）。
- android:verticalSpacing：用于设置各元素之间的垂直间距。

GridView 和 ListView 类似，都需要通过 Adapter 提供要显示的数据。在使用 GridView 控件时，通常使用 SimpleAdapter 或者 BaseAdapter 类 GridView 控件提供数据。

2. 网格视图实例

下面通过一个具体实例来演示 GridView 的用法。

【例 4-11】 使用一个 GridView 以行、列的形式来组织所有图片的预览视图，然后用一个 ImageSwitcher 来显示图片。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_11GridV。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中简单地定义一个 GridView、一个 ImageSwitcher。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
     android:orientation = "vertical"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent"
     android:gravity = "center_horizontal"
     android:background = "@drawable/bj1">
<! -- 定义一个 GridView 控件 -->
<GridView
     android:id = "@ + id/grid01"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:horizontalSpacing = "pt"
```

```
        android:verticalSpacing = "2pt"
        android:numColumns = "4"
        android:gravity = "center"/>
<! -- 定义一个 ImageSwitcher 控件 -->
< ImageSwitcher android:id = "@ + id/switcher"
        android:layout_width = "320dp"
        android:layout_height = "320dp"
        android:layout_gravity = "center_horizontal"/>
</LinearLayout >
```

(3) 选择 res\layout 目录,创建一个名为 cell.xml 的文件,该文件只包含一个简单的 ImageView 控件。代码为:

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "center_horizontal"
        android:padding = "5pt">
< ImageView
        android:id = "@ + id/image1"
        android:layout_width = "50dp"
        android:layout_height = "50dp" />
</LinearLayout >
```

(4) 打开 src\fs.li4_11gridv 目录下的 MainActivity.java 文件,程序使用 SimpleAdapter 对象作为 GridView 的内容适配器,这个 SimpleAdapter 底层保证了一个长度为 16 的 List 集合。代码为:

```
public class MainActivity extends Activity
{
    private static final String TAG = " == CrazyIt.org == ";
    int[] imageIds = new int[]
    {
        R.drawable.b1 , R.drawable.b2 , R.drawable.b3, R.drawable.b4,
        R.drawable.b5 , R.drawable.b6, R.drawable.b7 , R.drawable.b8
    };
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //创建一个 List 对象,List 对象的元素是 Map
        List < Map < String, Object >> listItems
            = new ArrayList < Map < String, Object >>();
        for (int i = 0; i < imageIds.length; i++)
        {
            Map < String, Object > listItem = new HashMap < String, Object >();
            listItem.put("image" , imageIds[i]);
            listItems.add(listItem);
        }
        //获取显示图片的 ImageSwitcher
        final ImageSwitcher switcher = (ImageSwitcher)
```

```
            findViewById(R.id.switcher);
        //设置图片更换的动画效果
        switcher.setInAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_in));
        switcher.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_out));
        //为 ImageSwitcher 设置图片切换的动画效果
        switcher.setFactory(new ViewFactory()
        {
            @Override
            public View makeView()
            {
                ImageView imageView = new ImageView(MainActivity.this);
                imageView.setBackgroundColor(0xff0000);
                imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
                imageView.setLayoutParams(new ImageSwitcher.LayoutParams(
                    LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
                return imageView;
            }
        });
        //创建一个 SimpleAdapter
        SimpleAdapter simpleAdapter = new SimpleAdapter(this, listItems
            //使用 layout\cell.xml 文件作为界面布局
            , R.layout.cell
            , new String[]{"image"}
            , new int[]{R.id.image1});
        GridView grid = (GridView)findViewById(R.id.grid01);
        //为 GridView 设置 Adapter
        grid.setAdapter(simpleAdapter);
        //添加列表项被选中的监听器
        grid.setOnItemSelectedListener(new OnItemSelectedListener()
        {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
                int position , long id)
            {
                //显示当前被选中的图片
                switcher.setImageResource(imageIds[position % imageIds.length]);
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent){}
        });
        //添加列表项被单击的监听器
        grid.setOnItemClickListener(new OnItemClickListener()
        {
            @Override
            public void onItemClick(AdapterView<?> parent,
                View view, int position, long id)
            {
                //显示被单击图片预览对应的图片
                switcher.setImageResource(imageIds[position % imageIds.length]);
            }
        });
    }
}
```

运行程序，在界面上显示了 8 个图片预览，单击任何一张图片预览，下面的 ImageSwitcher 都会显示对应的图片，效果如图 4-16 所示。



图 4-16　GridView 界面

4.4.3　图像切换器概述及实例

1. 图像切换器概述

图像切换器使用 ImageSwitcher 表示，用于实现类似于 Windows 操作系统下的“Windows 照片查看器”中的上一张、下一张切换图片的功能。在使用 ImageSwitcher 时，必须实现 ViewSwitcher. ViewFactory 接口，并通过 makeView()方法创建用于显示图片的 ImageView。makeView()方法将返回一个显示图片的 ImageView。在使用图像切换器时，还有一个方法非常重要，那就是 setImageResource()方法，该方法用于指定要在 ImageSwitcher 中显示的图片资源。

2. 图像切换器实例

下面通过一个实例来说明图像切换器的使用。

【例 4-12】　使用 ImageSwitcher 实现手机图片查看。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_12ImageSwitcher。

(2) 打开 res\layout 目录下的 main. xml 文件，在该文件中定义一个 ImageSwitcher、一个 Gallery。其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent">
    < ImageSwitcher android:id = "@ + id/switcher"
```

```
            android:layout_width = "match_parent"
            android:layout_height = "match_parent"
            android:layout_alignParentTop = "true"
            android:layout_alignParentLeft = "true"
            android:background = "@drawable/bj1"/>
        < Gallery android:id = "@ + id/gallery"
            android:background = "#55000000"
            android:layout_width = "match_parent"
            android:layout_height = "60dp"
            android:layout_alignParentBottom = "true"
            android:layout_alignParentLeft = "true"
            android:gravity = "center_vertical"
            android:spacing = "16dp"/>
</RelativeLayout >
```

（3）打开 src\fs. li4_12imageswitcher 目录下的 MainActivity. java 文件，用于实现图片的显示。代码为：

```
public class MainActivity extends Activity implements
 OnItemSelectedListener, ViewFactory {
 private ImageSwitcher is;
 private Gallery gallery;
 private Integer[] mThumbIds = { R.drawable.c1, R.drawable.c2,
   R.drawable.c3, R.drawable.c4, R.drawable.c5,R.drawable.c8,};
 private Integer[] mImageIds = { R.drawable.c1, R.drawable.c2,
   R.drawable.c3, R.drawable.c4, R.drawable.c5, R.drawable.c8,};
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 requestWindowFeature(Window.FEATURE_NO_TITLE);
 setContentView(R.layout.main);
 is = (ImageSwitcher) findViewById(R.id.switcher);
 is.setFactory(this);
 is.setInAnimation(AnimationUtils.loadAnimation(this,
    android.R.anim.fade_in));
 is.setOutAnimation(AnimationUtils.loadAnimation(this,
    android.R.anim.fade_out));
 gallery = (Gallery) findViewById(R.id.gallery);
 gallery.setAdapter(new ImageAdapter(this));
 gallery.setOnItemSelectedListener(this);
 }
 @Override
 public View makeView() {
 ImageView i = new ImageView(this);
 i.setBackgroundColor(0xFF000000);
 i.setScaleType(ImageView.ScaleType.FIT_CENTER);
 i.setLayoutParams(new ImageSwitcher.LayoutParams(
    LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT));
 return i;
 }
 public class ImageAdapter extends BaseAdapter {
 public ImageAdapter(Context c) {
   mContext = c;
 }
```

```
    public int getCount() {
      return mThumbIds.length;
    }
    public Object getItem(int position) {
      return position;
    }
    public long getItemId(int position) {
      return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
      ImageView i = new ImageView(mContext);
      i.setImageResource(mThumbIds[position]);
      i.setAdjustViewBounds(true);
      i.setLayoutParams(new Gallery.LayoutParams(
        LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
      i.setBackgroundResource(R.drawable.face);
      return i;
    }
    private Context mContext;
    }
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position,
      long id) {
    is.setImageResource(mImageIds[position]);
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
   }
```

运行程序，界面的默认效果如图 4-17 所示，单击下面任意一张图片，则对应的图片将显示在屏幕中间。



图 4-17 图像浏览

4.4.4 画廊视图属性及实例

1. 画廊视图

画廊视图使用 Gallery 表示，能够按水平方向显示内容，并且用户可用手指直接拖动图片移动。画廊视图一般用来浏览图片，被选中的选项位于中间，并且可以响应事件显示信息。在使用画廊视图时，首先需要在屏幕上添加 Gallery 控件，通常使用<Gallery>标记在 XML 布局文件中添加。在 XML 布局文件中添加画廊视图的格式为：

```
<Gallery
  属性列表>
</Gallery>
```

Gallery 控件支持的 XML 属性如下。

- android:animationDuration：用于设置列表项切换时的动画持续时间。
- android:gravity：用于设置对齐方式。
- android:spacing：用于设置列表项之间的间距。
- android:unselectedAlpha：用于设置没有选中的列表项的透明度。

使用画廊视图，也需要使用 Adapter 提供要显示的数据，通常使用 BaseAdapter 类为 Gallery 组件提供数据。

2. 画廊视图实例

下面通过一个具体实例演示画廊视图的操作。

【例 4-13】 通过 Gallery 循环显示图像，当单击某一个 Gallery 控件中的图像时在下方显示一个放大的图像(使用 ImageSwitcher 控件)。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_13Gallery。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中定义一个 Gallery 控件和一个 ImageSwitcher 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1">
<! -- 定义一个 ImageSwitcher 控件 -- >
<ImageSwitcher android:id = "@ + id/switcher"
    android:layout_width = "320dp"
    android:layout_height = "320dp"/>
<! -- 定义一个 Gallery 控件 -- >
<Gallery android:id = "@ + id/gallery"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:layout_marginTop = "25dp"
    android:unselectedAlpha = "0.6"
    android:spacing = "3pt"/>
</LinearLayout>
```

(3) 在 res\values 目录下创建一个名为 attrs.xml 的文件，用于定义画廊元素背景。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Gallery">
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>
```

(4) 打开 src\fs.li4_13gallery 目录下的 MainActivity.java 文件，为了让 ImageSwitcher 可显示 Gallery 中选中的图片，为 Gallery 绑定 OnItemSelectedListener 监听器。代码为：

```
public class MainActivity extends Activity
{
    int[] imageIds = new int[]
    {
        R.drawable.d1, R.drawable.d2,R.drawable.d3,
        R.drawable.d4, R.drawable.d5,R.drawable.d6,
        R.drawable.d7, R.drawable.d8};
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Gallery gallery = (Gallery) findViewById(R.id.gallery);
        //获取显示图片的 ImageSwitcher 对象
        final ImageSwitcher switcher = (ImageSwitcher)
            findViewById(R.id.switcher);
        //为 ImageSwitcher 对象设置 ViewFactory 对象
        switcher.setFactory(new ViewFactory()
        {
            @Override
            public View makeView()
            {
                ImageView imageView = new ImageView(MainActivity.this);
                imageView.setBackgroundColor(0xff0000);
                imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
                imageView.setLayoutParams(new ImageSwitcher.LayoutParams(
                    LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
                return imageView;
            }
        });
        //设置图片更换的动画效果
        switcher.setInAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_in));
        switcher.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_out));
        //创建一个 BaseAdapter 对象，该对象负责提供 Gallery 所显示的图片
        BaseAdapter adapter = new BaseAdapter()
        {
```

```
                @Override
                public int getCount()
                {
                    return imageIds.length;
                }
                @Override
                public Object getItem(int position)
                {
                    return position;
                }
                @Override
                public long getItemId(int position)
                {
                    return position;
                }
                //该方法返回的 View 代表了每个列表项
                @Override
                public View getView(int position, View convertView, ViewGroup parent)
                {
                    //创建一个 ImageView
                    ImageView imageView = new ImageView(MainActivity.this);
                    imageView
                        .setImageResource(imageIds[position % imageIds.length]);
                    //设置 ImageView 的缩放类型
                    imageView.setScaleType(ImageView.ScaleType.FIT_XY);
                    imageView.setLayoutParams(new Gallery.LayoutParams(75, 100));
                    TypedArray typedArray = obtainStyledAttributes(R.styleable.Gallery);
                    imageView. setBackgroundResource ( typedArray. getResourceId ( R. styleable.
                    Gallery_android_galleryItemBackground, 0));
                    return imageView;
                }
            };
            gallery.setAdapter(adapter);
            gallery.setOnItemSelectedListener(new OnItemSelectedListener()
            {
                //当 Gallery 选中项发生改变时触发该方法
                @Override
                public void onItemSelected(AdapterView<?> parent, View view,
                    int position, long id)
                {
                    switcher.setImageResource(imageIds[position % imageIds.length]);
                }
                @Override
                public void onNothingSelected(AdapterView<?> parent)
                {
                }
            });
        }
    }
```

运行程序,效果如图 4-18 所示。



图 4-18　画廊实例效果

4.5　其他控件

4.5.1　滚动视图概述及实例

1. 滚动视图概述

滚动视图用 ScrollView 表示，用于为其他组件添加滚动条。在默认情况下，当窗体中的内容比较多一屏幕显示不下时，超出的部分将不能被用户所看到，因为 Android 的布局管理器本身没有提供滚动屏幕的功能。如果要让其滚动，就需要使用滚动视图 ScrollView，这样用户就可以通过滚动屏幕来查看完整的内容了。

滚动视图是 android. widget. FrameLayout（帧布局管理器）的子类，因此，在滚动视图中可以添加任何想要放入其中的控件。但是，一个滚动视图中只能放置一个控件。如果想要放置多个控件，可以先放置一个布局管理器，再将要放置的其他控件放置到该布局管理器中。在滚动视图中，使用比较多的是线性布局管理器。

说明：滚动视图 ScrollView 只支持垂直滚动，如果想要实现水平滚动，可以使用水平滚动视图（HorizontalScrollView）。

在 Android 中可以使用两种方法向屏幕中添加滚动视图，一种是通过在 XML 布局文件中使用<ScrollView>标记添加，另一种是在 Java 文件中通过 new 关键字创建。

1）在 XML 布局文件中添加

在 XML 布局文件中添加滚动视图比较简单，只要在添加滚动条的控件外面使用下面的布局代码即可。

```
<ScrollView
    android:id="@+id/scrollView1"
```

```
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_below = "@ + id/textView1"
    android:layout_centerHorizontal = "true"
    android:layout_marginTop = "152dp" >
```

例如,要为一个显示公司简介的 TextView 添加滚动条,代码为:

```
<ScrollView
    android:id = "@ + id/scrollView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_below = "@ + id/textView1"
    android:layout_centerHorizontal = "true"
    android:layout_marginTop = "152dp" >
<TextView
    android:id = "@ + id/textView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "@string/content" />
</ScrollView>
```

2) 通过 new 关键字创建

在 Java 代码中,通过 new 关键字创建滚动视图比较简单,只需要经过以下步骤即可实现。

(1) 使用构造方法 ScrollView(Context context)创建一个滚动视图。

(2) 创建或者获取需要添加滚动条的控件,并应用 addView()方法将其添加到滚动视图中。

(3) 将滚动视图添加到整个布局管理器中,用于显示该滚动视图。

2. 滚动视图实例

下面通过实例来说明滚动视图的用法。

【例 4-14】 在屏幕上用滚动视图垂直显示多张图片。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_14ScrollView。

(2) 打开 res\layout 目录下的 main. xml 布局文件,在文件中定义一个 ScrollView 控件、10 个 ImageView。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<ScrollView
xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:scrollbars = "vertical">
    <LinearLayout android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:background = "@drawable/bj1">
        <ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
            android:layout_gravity = "center_horizontal"/>
        <ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
```

```
            android:layout_gravity = "center_horizontal"/>
        < ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
            android:layout_gravity = "center_horizontal"/>
        < ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
            android:layout_gravity = "center_horizontal"/>
        < ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
            android:layout_gravity = "center_horizontal"/>
        < ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
            android:layout_gravity = "center_horizontal"/>
        < ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
            android:layout_gravity = "center_horizontal"/>
        < ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
            android:layout_gravity = "center_horizontal"/>
        < ImageView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/face"
            android:layout_gravity = "center_horizontal"/>
    </LinearLayout >
</ScrollView >
```

运行程序,效果如图 4-19 所示,拖动鼠标即可浏览图片。



图 4-19　滚动视图效果

4.5.2 进度条属性及实例

1. 进度条属性

当一个应用程序在后台执行时，前台界面不会有任何信息，这时用户根本不知道程序是否在执行和执行进度等，因此需要使用进度条来提示程序执行的进度。在 Android 中，进度条使用 ProgressBar 表示，用于向用户显示某个耗时操作完成的百分比。

在屏幕中添加进度条，可以在 XML 布局文件中通过＜ProgressBar＞标记实现，格式为：

```
<ProgressBar
  属性列表>
</ProgressBar>
```

ProgressBar 控件支持的 XML 属性如下。

- android:max：用于设置进度条的最大值。
- android:progress：用于指定进度条已完成的进度值。
- android:progressDrawable：用于设置进度条轨道的绘制形式。

除了以上属性外，进度条控件还提供了以下两个常用方法用来操作进度。

- setProgress(int progress)方法：用于设置进度完成的百分比。
- incrementProgressBy(int diff)方法：用于设置进度条的进度增加或减少。当参数值为正数时表示进度增加，为负数时表示进度减少。

2. 进度条实例

下面通过一个实例来显示两种进度条。

【例 4-15】 实现两种进度条实例。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_15ProgressBar。

(2) 打开 res\layout 目录下的 main.xml 布局文件，布局一个 TextView 控件、两个 ProgressBar 控件和一个 Button 控件(圆形进度条和长形进度条这里样式不同于系统自带的)。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:background="@drawable/bj1">
     <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="欢迎 ProgressBar" />
        <!-- 定义一个水平进度条 -->
     <ProgressBar
         android:id="@+id/rectangleProgressBar"
         style="@android:style/Widget.ProgressBar.Horizontal"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
```

```
        android:visibility="gone" />
    <!-- 定义一个圆形进度条 -->
    <ProgressBar
        android:id="@+id/circleProgressBar"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"/>
    <Button android:id="@+id/button"
            android:text="显示进度"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
</LinearLayout>
```

(3) 打开 src\li4_15 progressbar 目录下的 MainActivity.java 文件，实现 Handler 和 Message 方法等。代码为：

```
public class MainActivity extends Activity {
    private ProgressBar rectangleProgressBar,circleProgressBar;
    private Button mButton;
    protected static final int STOP = 0x10000;
    protected static final int NEXT = 0x10001;
    private int iCount = 0;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);              //通过 id 查找视图
        rectangleProgressBar = (ProgressBar)findViewById(R.id.rectangleProgressBar);
        circleProgressBar = (ProgressBar)findViewById(R.id.circleProgressBar);
        mButton = (Button)findViewById(R.id.button);
        rectangleProgressBar.setIndeterminate(false);
        circleProgressBar.setIndeterminate(false);
        mButton.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                rectangleProgressBar.setVisibility(View.VISIBLE);
                circleProgressBar.setVisibility(View.VISIBLE);
                rectangleProgressBar.setMax(100);
                rectangleProgressBar.setProgress(0);
                circleProgressBar.setProgress(0);
                //创建一个线程,以每秒步长为 5 增加,到 100% 时停止
                Thread mThread = new Thread(new Runnable() {
                    public void run() {
                        for(int i=0 ; i<20; i++){
                            try{
                                iCount = (i + 1) * 5;
                                Thread.sleep(1000);
                                if(i == 19){
                                    Message msg = new Message();
                                    msg.what = STOP;
                                    mHandler.sendMessage(msg);
                                    break;
                                }else{
                                    Message msg = new Message();
                                    msg.what = NEXT;
                                    mHandler.sendMessage(msg);
```

```
                                }
                            }catch (Exception e) {
                                e.printStackTrace();
                            }
                        }
                    }
                });
                mThread.start();
            }
        });
    }
    //定义一个 Handler
    private Handler mHandler = new Handler(){
        public void handleMessage(Message msg){
            switch (msg.what) {
            case STOP:
                rectangleProgressBar.setVisibility(View.GONE);
                circleProgressBar.setVisibility(View.GONE);
                Thread.currentThread().interrupt();
                break;
            case NEXT:
                if(!Thread.currentThread().isInterrupted()){
                    rectangleProgressBar.setProgress(iCount);
                    circleProgressBar.setProgress(iCount);
                }
                break;
            }
        }
    };
}
```

运行程序，单击“显示进度”按钮，效果如图 4-20 所示。



图 4-20 进度条

4.5.3 拖动条概述及实例

1. 拖动条概述

拖动条和进度条类似，所不同的是，拖动条允许用户通过拖动滑块来改变值，通常用于实现对某种数值的调节。例如，调节图片的透明度或音量等。

在 Android 中，如果想在屏幕中添加拖动条，可在 XML 布局文件中通过<SeekBar>标记添加，格式为：

```
<SeekBar
    android:id = "@ + id/seekBar1"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:layout_alignParentLeft = "true"
    android:layout_below = "@ + id/textView1"
    android:layout_marginTop = "92dp" />
```

SeekBar 控件允许用户改变拖动滑块的外观，这可使用 android:thumb 属性实现，该属性的值为一个 Drawable 对象，该 Drawable 对象将作为自定义滑块。

由于拖动条可被用户控制，所以需要为其添加 OnSeekBarChangeListener 监听器。为拖动条添加监听器的代码为：

```
seekbar.setOnSeekBarChangeListener(new OnSeekBarChangeListener(){
   @Override
   public void onStopTrackingTouch(SeekBar seekBar){
     //要执行的代码
     }
     @Override
     public void onStartTrackingTouch(SeekBar seekBar){
       //要执行的代码
       }
       @Override
       public void onProgressChanged(SeekBar seekBar,int progress,boolean fromUser){
       //其他要执行的代码
       }
});
```

说明：在上面代码中，onProgressChanged()方法中的参数 progress 表示当前进度，即拖动条的值。

2. 拖动条实例

下面通过一个实例来演示拖动条的操作。

【例 4-16】 通过拖动条改变图片的透明度。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_16SeekBar。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在其中定义一个 ImageView 控件用于显示图片，定义一个 SeekBar 控件用于动态改变图片的透明度。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
```

```
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/bj1">
    <ImageView
        android:id="@+id/image"
        android:layout_width="fill_parent"
        android:layout_height="240px"
        android:src="@drawable/a05"/>
    <!-- 定义一个拖动条,并改变它的滑块外观,当前最大值为 255 -->
    <SeekBar
        android:id="@+id/seekbar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:max="255"
        android:progress="255"
        android:thumb="@drawable/ic_launcher"/>
    </LinearLayout>
```

(3) 打开 src\fs.li4_16seekback 目录下的 MainActivity.java 文件,为拖动条绑定一个监听器,当滑块位置发生改变时动态改变 ImageView 的透明度。代码为:

```
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final ImageView image = (ImageView)findViewById(R.id.image);
        SeekBar seekBar = (SeekBar)findViewById(R.id.seekbar);
        seekBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener()
        {
            //当拖动条的滑块位置发生改变时触发该方法
            @Override
            public void onProgressChanged(SeekBar arg0
                , int progress, boolean fromUser)
            {
                //动态改变图片的透明度
                image.setAlpha(progress);
            }
            @Override
            public void onStartTrackingTouch(SeekBar bar){}
            @Override
            public void onStopTrackingTouch(SeekBar bar){}
        });
    }
}
```

运行程序,效果如图 4-21 所示。



图 4-21　拖动条

4.5.4　星级评分条属性及实例

1. 星级评分条属性

星级评分条和拖动条类似,都允许用户通过拖动来改变进度,所不同的是,星级评分条通过星形表示进度。通常使用星级评分条表示对某一事物的支持度或对某种服务的满意程度等。例如,淘宝网中对卖家的好评度即通过星级评分条实现的。

在 Android 中,如果想在屏幕中添加星级评分条,可在 XML 布局文件中通过 <RatingBar>标记实现,格式为:

```
< RatingBar
  属性列表>
</ RatingBar >
```

RatingBar 控件支持的 XML 属性如下。

- android:isIndicator:用于指定该星级评分条是否允许用户改变,true 为不允许改变。
- android:numStars:用于指定该星级评分条共有多少个星。
- android:rating:用于指定该星级评分条默认的星级。
- android:stepSize:用于指定每次最少需要改变多少个星级,默认为 0.5。

除此之外,星级评分条还提供了 3 个比较常用的方法。

- getRating()方法:用于获取等级,表示被选中了几颗星。
- getStepSize()方法:用于获取每次最少要改变多少个星级。
- getProgress()方法:用于获取进度,获取到的进度值等于 getRating()方法的返回值乘以 getStepSize()方法的返回值。

2. 星级评分条实例

下面通过一个实例来说明星级评分条的用法。

【例 4-17】 利用 RatingBar 控件实现评分。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_17RatingBar。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中定义一个 TextView 控件、3 个 RatingBar 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1">
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "你得到的星个数" />
< RatingBar
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    style = "?android:attr/ratingBarStyleIndicator"
    android:id = "@ + id/ratingbar_Indicator" />
< RatingBar
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    style = "?android:attr/ratingBarStyleSmall"
    android:id = "@ + id/ratingbar_Small"
    android:numStars = "20" />
< RatingBar
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    style = "?android:attr/ratingBarStyle"
    android:id = "@ + id/ratingbar_default" />
</LinearLayout >
```

(3) 打开 src\fs. li4_17ratingbar 目录下的 MainActivity. java 文件，为 RatingBar 绑定事件监听器，监听星级评分条的星级改变。代码为：

```
public class MainActivity extends Activity {
    /* 第一次调用 Activity  */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
      final RatingBar ratingBar_Small = (RatingBar)findViewById(R.id.ratingbar_Small);
  final RatingBar ratingBar_Indicator = (RatingBar)findViewById(R.id.ratingbar_Indicator);
    final RatingBar ratingBar_default = (RatingBar)findViewById(R.id.ratingbar_default);
            ratingBar _ default. setOnRatingBarChangeListener ( new RatingBar.
OnRatingBarChangeListener(){
    public void onRatingChanged(RatingBar ratingBar, float rating,
```

```
     boolean fromUser) {
    ratingBar_Small.setRating(rating);
    ratingBar_Indicator.setRating(rating);
    Toast.makeText(MainActivity.this,"你得到了" + rating + "颗星",Toast.LENGTH_LONG).show();
   }});
   }
}
```

运行程序,单击星形,即可显示所得到的星个数,效果如图 4-22 所示。



图 4-22 星级评分条

4.6 时间类控件

在 Android 中提供了一些与日期和时间相关的控件,常用的控件有日期选择器、时间选择器和计时器等。

4.6.1 日期、时间控件概述及实例

1. 日期、时间控件概述

为了让用户选择日期和时间,Android 提供了日期、时间选择器,分别对应 DatePicker 和 TimePicker 控件。这两个控件的使用比较简单,可以在 Eclipse 的可视化界面设计器中选择对应的控件将其拖到布局文件中。为了在程序中获取用户选择的日期、时间,还需要为 DatePicker 和 TimePicker 控件添加事件监听器。其中,DatePicker 控件对应的事件监听器是 OnDateChangedListener,TimePicker 控件对应的事件监听器是 OnTimeChangedListener。

2. 日期、时间控件实例

下面通过一个实例来说明日期、时间选择器的具体用法。

【例 4-18】 在界面中实现日期、时间的选择。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_18DateTime。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中定义一个 DatePicker 控件、一个 TimePicker 控件和两个 EditText 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
       android:orientation = "vertical"
       android:layout_width = "fill_parent"
       android:layout_height = "fill_parent"
       android:background = "@drawable/bj1">
< DatePicker android:id = "@ + id/datePicker"
       android:layout_width = "wrap_content"
       android:layout_height = "wrap_content"
       android:layout_gravity = "center_horizontal"/>
< EditText android:id = "@ + id/dateEt"
       android:layout_width = "fill_parent"
       android:layout_height = "wrap_content"
       android:cursorVisible = "false"
       android:editable = "false"/>
< TimePicker android:id = "@ + id/timePicker"
       android:layout_width = "wrap_content"
       android:layout_height = "wrap_content"
       android:layout_gravity = "center_horizontal"/>
< EditText android:id = "@ + id/timeEt"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:cursorVisible = "false"
     android:editable = "false"/>
</LinearLayout >
```

(3) 打开 src\fs.li4_18datetime 目录下的 MainActivity.java 文件，实现时间与日期的选择。代码为：

```
public class MainActivity extends Activity {
    private EditText dateEt = null;
    private EditText timeEt = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        dateEt = (EditText)findViewById(R.id.dateEt);
        timeEt = (EditText)findViewById(R.id.timeEt);
        //获取日期拾取控件和时间拾取控件
        DatePicker datePicker = (DatePicker)findViewById(R.id.datePicker);
        TimePicker timePicker = (TimePicker)findViewById(R.id.timePicker);
        //创建一个日历对象
        Calendar calendar = Calendar.getInstance();
        int year = calendar.get(Calendar.YEAR);                    //获取当前年份
        int monthOfYear = calendar.get(Calendar.MONTH);            //获取当前月份
        int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);      //获取当前日期
```

```
        //初始化日期器,并在初始化时指定监听器
        datePicker.init(year, monthOfYear, dayOfMonth, new OnDateChangedListener(){
public void onDateChanged(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
dateEt.setText("您选择的日期是: " + year + "年" + (monthOfYear + 1) + "月" + dayOfMonth +
"日.");
                }
            });
        timePicker.setOnTimeChangedListener(new OnTimeChangedListener(){
            public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            timeEt.setText("您选择的时间是: " + hourOfDay + "时" + minute + "分.");
                }
            });
        }
    }
```

运行程序,效果如图 4-23 所示。



图 4-23　时间与日期的选择

4.6.2　时钟控件概述及实例

1. 时钟控件概述

时钟 UI 控件包括两个非常简单的控件,DigitalClock 本身就继承了 TextView,也就是说其本身就是文本框,只是它里面显示的内容是当前时间;AnalogClock 则继承了 View 控件,其重写了 View 的 OnDraw 方法,会在 View 上显示模拟时钟。

DigitalClock 和 AnalogClock 都会显示当前时间。不同的是,DigitalClock 显示数字时钟,可以显示当前的秒数;而 AnalogClock 显示模拟时钟,不会显示当前秒数。

2. 时钟控件实例

下面通过实例来演示 AnalogClock 怎样实现模拟时钟。

【例 4-19】 实现模拟时钟。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_19ADClock。

(2) 打开 res\layout 目录下的 main.xml 布局文件，布局一个文本框和一个模拟时钟控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
  android:id = "@ + id/widget27"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent"
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:orientation = "vertical"
  android:background = "@drawable/bj1">
<AnalogClock
  android:id = "@ + id/myAnalogClock"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_gravity = "center_horizontal">
</AnalogClock>
<TextView
  android:id = "@ + id/myTextView"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:text = "TextView"
  android:textSize = "20sp"
  android:textColor = "@drawable/white"
  android:layout_gravity = "center_horizontal">
</TextView>
</LinearLayout>
```

(3) 在 res\values 目录中创建一个名为 color.xml 的文件，用于实现文本框的颜色。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
  <drawable name = "white">#FF1F0F</drawable>
</resources>
```

(4) 打开 src\fs.li4_19adclock 目录下的 MainActivity.java 文件，实现时钟组件。代码为：

```
public class MainActivity extends Activity
{
 /* 声明一个常数作为判别信息 */
 protected static final int GUINOTIFIER  =  0x1234;
 /* 声明两个 widget 对象变量 */
 private TextView mTextView;
 public AnalogClock mAnalogClock;
 /* 声明与时间相关的变量 */
 public Calendar mCalendar;
 public int mMinutes;
```

```
public int mHour;
/ * 声明 Handler 和 Thread 变量 * /
public Handler mHandler;
private Thread mClockThread;
/ *  第一次调用活动  * /
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    / * 通过 findViewById 取得两个 widget 对象 * /
    mTextView = (TextView)findViewById(R.id.myTextView);
    mAnalogClock = (AnalogClock)findViewById(R.id.myAnalogClock);
    / * 通过 Handler 接收运行线程所传递的信息并更新 TextView * /
    mHandler  =  new Handler()
    {
      public void handleMessage(Message msg)
      {
        / * 这里是处理信息的方法 * /
        switch (msg.what)
        {
          case MainActivity.GUINOTIFIER:
          / *  在这里处理 TextView 对象 Show 时间的事件 * /
            mTextView.setText(mHour + "  :  " + mMinutes);
            break;
        }
        super.handleMessage(msg);
      }
    };
    / * 通过运行线程持续取得系统时间 * /
     mClockThread = new LooperThread();
     mClockThread.start();
}
/ * 改写一个 Thread Class 用来持续取得系统时间 * /
class LooperThread extends Thread
{
    public void run()
    {
      super.run();
      try
      {
        do
        {
          / * 取得系统时间 * /
          long time  =  System.currentTimeMillis();
          / * 通过 Calendar 对象取得小时与分钟 * /
          final Calendar mCalendar  =  Calendar.getInstance();
          mCalendar.setTimeInMillis(time);
          mHour  =  mCalendar.get(Calendar.HOUR);
          mMinutes  =  mCalendar.get(Calendar.MINUTE);
          / * 让运行线程休息一秒 * /
          Thread.sleep(1000);
            / * 关键程序:取得时间后发出信息给 Handler * /
            Message m  =  new Message();
            m.what  =  MainActivity.GUINOTIFIER;
```

```
                MainActivity.this.mHandler.sendMessage(m);
            }while(MainActivity.LooperThread.interrupted() == false);
            /* 当系统发出中断信息时停止本循环 */
          }
    catch(Exception e)
          {
            e.printStackTrace();
          }
        }
  }
}
```

运行程序，效果如图 4-24 所示。



图 4-24　时钟实例效果

4.6.3　计时器概述及实例

1. 计时器概述

计时器控件可实现显示从某个起始时间开始一共过去了多长时间的文本，使用 Chronometer 表示。由于该控件继承自 TextView，所以它将以文本的形式显示内容。该控件比较简单，通常只需要使用以下 5 个方法。

- setBase()：用于设置计时器的起始时间。
- setFormat()：用于设置显示时间的格式。
- start()：用于指定开始计时。
- stop()：用于指定停止计时。
- setOnChronometerTickListener()：用于为计时器绑定事件监听器，当计时器改变时触发该监听器。

说明：在默认情况下，计时器返回的值为 MM:SS 的格式，例如 8 分零 12 秒将显示为

08:12 的形式。在使用 setFormat()方法设置显示时间的格式时，可以使用%s 表示计时信息，例如要设置显示时间的格式为"已用时间：MM:SS"，可以将 setFormat()的参数设置为"已用时间：%s"。

2. 计时器实例

下面通过一个实例来讲解计时器的用法。

【例 4-20】 使用 Chronometer 控件实现 Android 计时器。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_20Chronometer。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中修改代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent"
      android:background = "@drawable/bj1"
      android:gravity = "center"
      android:orientation = "vertical">
< LinearLayout
     android:layout_width = "fill_parent"
      android:layout_height = "wrap_content"
      android:layout_margin = "10dip"
       android:orientation = "horizontal">
< TextView
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:layout_weight = "4"
      android:gravity = "center"
      android:text = "设置时间:"/>
< EditText
     android:id = "@ + id/edt_settime"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
      android:layout_weight = "1"
       android:inputType = "number"/>
</LinearLayout >
< Chronometer
     android:id = "@ + id/chronometer"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:gravity = "center"
     android:textColor = "#ff0000"
     android:textSize = "60dip"/>
< LinearLayout
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:layout_margin = "10dip"
     android:orientation = "horizontal">
< Button android:id = "@ + id/btnStart"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:layout_weight = "1"
```

```
        android:text = "开始计时"/>
    < Button android:id = "@ + id/btnStop"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_weight = "1" android:text = "停止计时"/>
    < Button android:id = "@ + id/btnReset"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_weight = "1"
        android:text = "重置"/>
    </LinearLayout >
    </LinearLayout >
```

(3) 打开 res\fs. li4_20chronometer 目录下的 MainActivity. java 文件，通过设置 setBase(long base)来设置初始时间，然后为其添加一 个 setOnChronometerTickListener (Chronometer. OnChronometerTickListener l)事件来判断时间是否到了，再调用其 stop() 方法实现停止计时。代码为：

```
public class MainActivity extends Activity {
    private int startTime = 0;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    final Chronometer chronometer = (Chronometer) findViewById(R.id.chronometer);
        Button btnStart = (Button) findViewById(R.id.btnStart);
        Button btnStop = (Button) findViewById(R.id.btnStop);
        Button btnRest = (Button) findViewById(R.id.btnReset);
        final EditText edtSetTime = (EditText) findViewById(R.id.edt_settime);
        btnStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                System.out.println(" -- 开始计时 --- ");
                String ss = edtSetTime.getText().toString();
                if (!(ss.equals("") && ss != null)) {
                    startTime = Integer.parseInt(edtSetTime.getText().toString());
                }
                //设置开始计时时间
                chronometer.setBase(SystemClock.elapsedRealtime());
                //开始计时
                chronometer.start();
                }
            });
        btnStop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //停止计时
                chronometer.stop();
                }
            });
        //重置
        btnRest.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                chronometer.setBase(SystemClock.elapsedRealtime());
                }
            });
chronometer.setOnChronometerTickListener(new Chronometer.OnChronometerTickListener() {
            @Override
            public void onChronometerTick(Chronometer chronometer) {
                //如果从开始计时到现在超过了 startime 秒
                if (SystemClock.elapsedRealtime()
                        - chronometer.getBase()> startTime * 1000) {
                    chronometer.stop();
                    //给用户提示
                    showDialog();
                    }
                }
            });
        }
    protected void showDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setIcon(R.drawable.ic_launcher);
        builder.setTitle("警告").setMessage("时间到").setPositiveButton("确定", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) { }
            });
        AlertDialog dialog = builder.create();
        dialog.show();
    }
}
```

运行程序,效果如图 4-25 所示。在“设置时间:”后面的编辑框中输入对应的时间,单击“开始计时”按钮,即可进行计时,效果如图 4-26 所示;当单击“重置”按钮时,计时器重新计时;当单击“停止计时”按钮时,计时停止;当计时完成时,会弹出相应的警告对话框,如图 4-27 所示。



图 4-25　计时器初始界面



图 4-26　开始计时



图 4-27　完成计时

4.7　基本控件综合实例

在本节中，将通过两个综合实例来演示各控件的应用。

4.7.1　体重器界面

前面已经介绍了文本框、按钮、选择框等多种控件，下面使用多种控件来实现体重器界面的设计。

其实现步骤如下：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li4_21WeightControl。

（2）打开 res\layout 目录下的 main.xml 布局文件，在布局文件中定义 4 个 TextView 控件、一个 EditText 控件、一个 RadioGroup 控件、两个 RadioButton 控件及一个 Button 控件。代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical"
    tools:context = ".MainActivity"
    android:background = "@drawable/bj1">
    <TextView
        android:id = "@ + id/showtext"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center"
```

```
        android:text = "@string/soft_name"
        android:textSize = "25sp" />
    < TextView
        android:id = "@ + id/text_Height"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/height" />
    < EditText
        android:id = "@ + id/height_Edit"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center"
        android:layout_weight = "0.05"
        android:ems = "10"
        android:hint = "@string/htext"
        android:textSize = "18sp" >
        < requestFocus />
    </EditText >
    < TextView
        android:id = "@ + id/text_sex"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center"
        android:text = "@string/sex" />
    < RadioGroup
        android:id = "@ + id/radiogroup"
        android:layout_width = "wrap_content"
        android:layout_height = "37px"
        android:layout_gravity = "center"
        android:orientation = "horizontal" >
        < RadioButton
            android:id = "@ + id/Sex_Man"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "@string/man" />
        < RadioButton
            android:id = "@ + id/Sex_Woman"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "@string/woman" />
    </RadioGroup >
    < Button
        android:id = "@ + id/button_ok"
        android:layout_width = "100dp"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center"
        android:text = "@string/ok" />
</LinearLayout >
```

(3) 打开 res\values 目录下的 strings. xml 文件，修改代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name">计重器界面</string >
```

```
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="soft_name">计算你的标准体重</string>
    <string name="height">身高:</string>
    <string name="htext">请输入身高</string>
    <string name="sex">性别:</string>
    <string name="man">男</string>
    <string name="woman">女</string>
    <string name="ok">计算</string>
</resources>
```

(4) 打开 MainActivity.java 文件,代码不做任何修改。

运行程序,效果如图 4-28 所示。



图 4-28 计重器界面

4.7.2 登录界面

登录界面对于绝大部分应用来说都是有必要的,下面通过实现一个记住密码的界面来综合掌握常用的控件。其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_22Login。

(2) 打开 res\layout 目录下的 main.xml 布局文件,用于实现登录主界面。代码为:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bj1"
    android:orientation="vertical" >
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
```

```
<ImageButton
    android:id="@+id/img_btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:background="@drawable/q1"/>
<TextView
    android:id="@+id/tv_zh"
    android:layout_width="wrap_content"
    android:layout_height="35dip"
    android:layout_marginLeft="12dip"
    android:layout_marginTop="10dip"
    android:gravity="bottom"
    android:text="账号:"
    android:textColor="#000000"
    android:textSize="18sp" />
<EditText
    android:id="@+id/et_zh"
    android:layout_width="fill_parent"
    android:layout_height="40dip"
    android:layout_below="@id/tv_zh"
    android:layout_marginLeft="12dip"
    android:layout_marginRight="10dip" />
<TextView
    android:id="@+id/tv_mima"
    android:layout_width="wrap_content"
    android:layout_height="35dip"
    android:layout_below="@id/et_zh"
    android:layout_marginLeft="12dip"
    android:layout_marginTop="10dip"
    android:gravity="bottom"
    android:text="密码:"
    android:textColor="#000000"
    android:textSize="18sp" />
<EditText
    android:id="@+id/et_mima"
    android:layout_width="fill_parent"
    android:layout_height="40dip"
    android:layout_below="@id/tv_mima"
    android:layout_marginLeft="12dip"
    android:layout_marginRight="10dip"
    android:maxLines="200"
    android:password="true"
    android:scrollHorizontally="true" />
<CheckBox
    android:id="@+id/cb_mima"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/et_mima"
    android:layout_marginLeft="12dip"
    android:text="记住密码"
    android:textColor="#000000" />
<CheckBox
    android:id="@+id/cb_auto"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/cb_mima"
            android:layout_marginLeft="12dip"
            android:text="自动登录"
            android:textColor="#000000" />
        <Button
            android:id="@+id/btn_login"
            android:layout_width="80dip"
            android:layout_height="40dip"
            android:layout_below="@id/et_mima"
            android:layout_alignParentRight="true"
            android:layout_alignTop="@id/cb_auto"
            android:layout_marginRight="10dip"
            android:gravity="center"
            android:text="登录"
            android:textColor="#000000"
            android:textSize="18sp"/>
    </RelativeLayout>
</LinearLayout>
```

(3) 在 res\layout 目录下创建一个名为 logo.xml 的文件,用于布局进度条和取消界面。代码为:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bj1"
    android:orientation="vertical" >
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="3">
        <ProgressBar
            android:id="@+id/pgBar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true" />
        <TextView
            android:id="@+id/tv1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/pgBar"
            android:layout_centerHorizontal="true"
            android:text="正在登录..."
            android:textColor="#000000"
            android:textSize="18sp" />
    </RelativeLayout>
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:orientation="vertical" >
```

```
        <Button
            android:id = "@ + id/btn_back"
            android:layout_width = "70dip"
            android:layout_height = "35dip"
            android:text = "取消"
            android:textColor = "#000000"
            android:textSize = "12sp" />
    </LinearLayout>
</LinearLayout>
```

(4) 在 res\layout 目录下创建一个名为 wel.xml 的文件,用于实现欢迎界面。代码为:

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:layout_gravity = "center"
    android:background = "@drawable/bj1"
    android:orientation = "vertical" >
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:gravity = "center"
        android:text = "登录成功,进入用户界面"
        android:textColor = "#000000"
        android:textSize = "20sp" />
</LinearLayout>
```

(5) 打开 src\fs.li4_22login 目录下的 MainActivity.java 文件,用于设置登录账号、密码等,实现账号登录功能。代码为:

```
public class MainActivity extends Activity {
    private EditText userName, password;
    private CheckBox rem_pw, auto_login;
    private Button btn_login;
    private ImageButton btnQuit;
    private String userNameValue,passwordValue;
    private SharedPreferences sp;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //去除标题
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);
        //获得实例对象
        sp = this.getSharedPreferences("userInfo", Context.MODE_WORLD_READABLE);
        userName = (EditText) findViewById(R.id.et_zh);
        password = (EditText) findViewById(R.id.et_mima);
        rem_pw = (CheckBox) findViewById(R.id.cb_mima);
        auto_login = (CheckBox) findViewById(R.id.cb_auto);
        btn_login = (Button) findViewById(R.id.btn_login);
        btnQuit = (ImageButton)findViewById(R.id.img_btn);
        //判断记住密码多选框的状态
      if(sp.getBoolean("ISCHECK", false))
        {
          //设置默认是记录密码状态
```

```
            rem_pw.setChecked(true);
            userName.setText(sp.getString("USER_NAME", ""));
            password.setText(sp.getString("PASSWORD", ""));
            //判断自动登录多选框状态
            if(sp.getBoolean("AUTO_ISCHECK", false))
            {
                  //设置默认是自动登录状态
                  auto_login.setChecked(true);
                //跳转界面
                Intent intent = new Intent(MainActivity.this,LogoActivity.class);
                MainActivity.this.startActivity(intent);
            }
        }
        //登录监听事件,现在默认用户名为 liu、密码为 123
        btn_login.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                userNameValue = userName.getText().toString();
                passwordValue = password.getText().toString();
                if(userNameValue.equals("MrZhange")&&passwordValue.equals("123a"))  {
                    Toast.makeText(MainActivity.this,"登录成功", Toast.LENGTH_SHORT).show();
                    //登录成功和记住密码框为选中状态才保存用户信息
                    if(rem_pw.isChecked())
                    {
                     //记住用户名、密码
                      Editor editor = sp.edit();
                      editor.putString("USER_NAME", userNameValue);
                      editor.putString("PASSWORD",passwordValue);
                      editor.commit();
                    }
                    //跳转界面
                    Intent intent = new Intent(MainActivity.this,LogoActivity.class);
                    MainActivity.this.startActivity(intent);
                    //finish();
                }else{
                      Toast.makeText(MainActivity.this,"用户名或密码错误,请重新登录",
Toast.LENGTH_LONG).show();
                }
            }
        });
        //监听记住密码多选框按钮事件
        rem_pw.setOnCheckedChangeListener(new OnCheckedChangeListener() {
     public void onCheckedChanged(CompoundButton buttonView,boolean isChecked) {
                if (rem_pw.isChecked()) {
                    System.out.println("记住密码已选中");
                    sp.edit().putBoolean("ISCHECK", true).commit();
                }else {
                    System.out.println("记住密码没有选中");
                    sp.edit().putBoolean("ISCHECK", false).commit();

                }
            }
        });
        //监听自动登录多选框事件
        auto_login.setOnCheckedChangeListener(new OnCheckedChangeListener() {
```

```
        public void onCheckedChanged(CompoundButton buttonView,boolean isChecked) {
                if (auto_login.isChecked()) {
                    System.out.println("自动登录已选中");
                    sp.edit().putBoolean("AUTO_ISCHECK", true).commit();
                } else {
                    System.out.println("自动登录没有选中");
                    sp.edit().putBoolean("AUTO_ISCHECK", false).commit();
                }
            }
        });
        btnQuit.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }
}
```

(6) 在 src\fs.li4_22login 目录下创建一个名为 LogoActivity.java 的文件,用于实现登录界面的退出。代码为:

```
public class LogoActivity extends Activity {
    private ProgressBar progressBar;
    private Button backButton;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //去除标题
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.logo);
        progressBar = (ProgressBar) findViewById(R.id.pgBar);
        backButton = (Button) findViewById(R.id.btn_back);
        Intent intent = new Intent(this, WelActivity.class);
        LogoActivity.this.startActivity(intent);
        backButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }
}
```

(7) 在 src\fs.li4_22login 目录下创建一个名为 WelActivity.java 的文件,用于实现欢迎界面。代码为:

```
public class WelActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.wel);
    }
}
```

运行程序,效果如图 4-29 所示。在界面中输入正确的账号和密码,即可登录界面;如果账号或密码错误,将弹出错误信息提示,效果如图 4-30 所示。




图 4-29 记住密码的登录界面



图 4-30 密码或账号错误提示

4.7.3 人物评分

所谓人物评分,就是选择人物并对该人物进行评分。下面通过一个例子来演示基本控件的综合使用,实现人物评分。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_23 Ratings。

(2) 将字符串声明到一个文件中,以便于系统的管理与维护。打开 res\values 目录下的 strings. xml 文件,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name">人物评分</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
    < string name = "menu_settings"> Settings </string >
    < string name = "show_text">请选择人物</string >
    < string name = "ping">请给人物评分</string >
    < string name = "g1">阿狸</string >
    < string name = "g2">九尾妖狐</string >
    < string name = "g3">潘斯特</string >
    < string name = "g4">悠嘻猴</string >
    < string name = "g5">洋葱头</string >
</resources >
```

(3) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明两个 TextView 控件、一个 Spinner 控件、一个 ImageView 控件和一个 RatingBar 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    tools:context = ".MainActivity"
    android:background = "@drawable/bj1" >
    <TextView
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/show_text" />
    <Spinner
        android:id = "@ + id/spinner1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentRight = "true"
        android:layout_below = "@ + id/textView1" />
    <ImageView
        android:id = "@ + id/imageView1"
        android:layout_width = "200px"
        android:layout_height = "200px"
        android:layout_centerHorizontal = "true"
        android:layout_centerVertical = "true" />
    <RatingBar
        android:id = "@ + id/ratingBar1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentLeft = "true" />
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_above = "@ + id/ratingBar1"
        android:layout_alignParentLeft = "true"
        android:text = "@string/ping" />
</RelativeLayout>
```

(4) 打开 res\fs.li4_23ratings 目录下的 MainActivity.java 文件，代码为：

```
public class MainActivity extends Activity {
    Spinner sp;
    ImageView iview;
    RatingBar rb;
    private ArrayAdapter<String> adapter;
    //人物描述
    //人物图片，初始化所有图片资源 ID
    int[] draws = { R.drawable.g1, R.drawable.g2, R.drawable.g3, R.drawable.g4,
```

```
            R.drawable.g5 };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //初始化下拉列表的显示数据
        String[] items = { this.getString(R.string.g1),
                this.getString(R.string.g2), this.getString(R.string.g3),
                this.getString(R.string.g4), this.getString(R.string.g5) };
        //初始化所有控件
        sp = (Spinner) findViewById(R.id.spinner1);
        iview = (ImageView) findViewById(R.id.imageView1);
        rb = (RatingBar) findViewById(R.id.ratingBar1);
        //内容适配器的初始化代码,使用了 Android 默认定义的布局方式和预显示内容
        adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_spinner_item, items);
        //定义下拉列表的选择事件处理,当下拉选择发生改变时将改变相应的图片显示内容
        sp.setAdapter(adapter);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        sp.setOnItemSelectedListener(new Spinner.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                    int arg2, long arg3) {
                //TODO 自动生成的方法存根
                iview.setImageResource(draws[arg2]);
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                //TODO 自动生成的方法存根
            }
        });
rb.setOnRatingBarChangeListener(new OnRatingBarChangeListener() {
            @Override
            public void onRatingChanged(RatingBar ratingBar, float rating,
                    boolean fromUser) {
                //TODO 自动生成的方法存根
                Toast.makeText(MainActivity.this, "你的评分为" + rating,
                        Toast.LENGTH_LONG).show();
            }
        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //增加了项目操作栏
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
 }
```

运行程序，默认界面如图 4-31(a)所示，通过单击进行选择，并给该人物评分后的效果如图 4-31(b)所示，对人物进行选择如图 4-31(c)所示。



(a) 默认界面




(b) 人物选择



(c) 人物评分

图 4-31　人物评分效果图

第5章 Android 菜单与对话框

5.1 菜　　单

菜单在桌面应用中的使用十分广泛,几乎所有的桌面应用都有菜单。现在,虽然菜单在手机应用中的使用减少了不少(主要受到手机屏幕大小的制约),但依然有不少手机应用会添加菜单。

与桌面应用的菜单不同,Android 应用中的菜单默认是看不见的,只有当用户按下手机上的 Menu 键(位于模拟器右边的物理键盘上)时,系统才会显示该应用关联的菜单。

Android 应用同样支持上下文菜单(ContextMenu),当用户一直按住某个应用的界面时,该应用所关联的上下文菜单就会显示出来。

5.1.1 菜单选项概述及实例

1. 菜单选项概述

当 Activity 在前台运行时,如果用户按下手机上的 Menu 键,就会在屏幕底端弹出相应的选项菜单。但这个功能是需要开发人员编程来实现的,如果在开发应用程序时没有实现该功能,那么在程序运行时按下手机上的 Menu 键将不会起作用。

对于带图标的选项菜单,每次最多只能显示 6 个,当菜单选项多于 6 个时,将只显示前 5 个和一个扩展菜单选项,单击扩展菜单选项将弹出其余菜单项。扩展菜单项中不会显示图标,但是可以显示单选按钮及复选框。

在 Android 中通过回调方法创建菜单并处理菜单项的按下事件,这些回调方法如下。

- onCreateOptionsMenu(Menu menu):初始化选项菜单,该方法只在第一次显示菜单时调用,如果需要每次显示菜单时更新菜单项,则需要重写 onPrepareOptionsMenu(Menu)方法。
- public boolean onOptionsItemSeleted(MenuItem item):当选项菜单中的某个选项被选中时调用该方法,默认是一个返回 false 的空实现。
- public void onOptionsMenuClosed(Menu menu):当选项菜单关闭时(或者由于用户按下了返回键或选择了某个菜单选项)调用该方法。
- public boolean onPrepareOptionsMenu(Menu menu):为程序准备选项菜单,每次选项菜单显示前会调用该方法,可以通过该方法设置某些菜单项可用或不可用或修改菜单项的内容。重写该方法时需要返回 true,否则选项菜单将不会显示。

提示: 除了可以使用回调方法 onOptionsItemSelected 来处理用户选中的菜单事件外,

还可以为每个菜单项对象 MenuItem 添加 OnMenuItemClickListener 监听器来处理菜单选中事件。

开发选项菜单主要用到 Menu、MenuItem 和 SubMenu，下面进行介绍。

1）Menu 类

一个 Menu 对象代表一个菜单，在 Menu 对象中可以添加菜单项 MenuItem，也可以添加子菜单 SubMenu。在 Menu 中常用的方法如表 5-1 所列。

表 5-1　Menu 的常用方法及说明

方法名称	参数说明	方法说明
MenuItem add(int groupId, int itemId, int order, CharSequence title); MenuItem add(int groupId, int itemId, int order, int titleRes); MenuItem add(CharSequence title); MenuItem add(int titleRes)	groupId 为菜单项所在的组 id，通过分组可以对菜单项进行批量操作，如果菜单项不需要属于任何组，则传入 NONE； itemId 为唯一标识菜单项的 id，可传入 NONE； order 为菜单项的顺序，可以传入 NONE；title 为菜单项显示的文本内容； titleRes 为 String 对象的资源标识符	向 Menu 添加一个菜单项，返回 MenuItem 对象
SubMenu addSubMenu(int titleRes); SubMenu addSubMenu(int groupId, int itemId, int order, int titleRes); SubMenu addSubMenu(Char-Sequence title); SubMenu addSubMenu(int groupId, int itemId, int order, CharSequence title)	groupId 为子菜单所在的组 id，通过分组可以对子菜单进行批量操作，如果子菜单不需要属于任何组，则传入 NONE； itemId 为唯一标识子菜单的 id，可传入 NONE； order 为子菜单的顺序，可传入 NONE； title 为子菜单显示的文本内容； titleRes 为 String 对象的资源标识符	向 Menu 添加一个子菜单，返回 SubMenu 对象
void clear()	…	移除菜单中所有的子项
void close()	…	如果菜单正在显示，则关闭菜单
MenuItem findItem(int id)	id 为 MenuItem 的标识符	返回指定 id 的 MenuItem 对象
void removeGroup(int groupId)	groupId 为组 id	如果指定 id 的组不为空，则从菜单中移除该组
void removeItem(int id)	id 为 MenuItem 的标识符	移除指定 id 的 MenuItem
int size()	…	返回 Menu 中菜单项的个数

2）MenuItem 对象

MenuItem 对象代表一个菜单项，通常 MenuItem 实例通过 Menu 的 add 方法获得。在 MenuItem 中常用的成员方法如表 5-2 所列。

表 5-2　选项菜单相关的回调方法及说明

方 法 名 称	参 数 说 明	方 法 说 明
setAlphabeticShortcut(char alphaChar)	alphaChar 为字母快捷键	设置 MenuItem 的字母快捷键
MenuItem setNumericShortcut(char numericChar)	numericChar 为数字快捷键	设置 MenuItem 的数字快捷键
MenuItem setIcon(Drawable icon)	icon 为图标 Drawable 对象	设置 MenuItem 的图标
MenuItem setIntent(Intent intent)	intent 为与 MenuItem 绑定的 Intent 对象	为 MenuItem 绑定 Intent 对象，当被选中时，将会调用 startActivity 方法处理相应的 Intent
setOnMenuItemClickListener (MenuItem. OnMenuItemClick-Listener menuItemClickListener)	menuItemClickListener 为监听器	为 MenuItem 设置自定义的监听器，一般情况下，使用回调方法 onOptionsItemSelected 会更有效率
setShortcut (char numericChar, char alphaChar)	numericChar 为数字快捷键；alphaChar 为字母快捷键	为 MenuItem 设置数字快捷键和字母快捷键，当按下快捷键或按住 Alt 的同时按下快捷键时将会触发 MenuItem 的选中事件
setTitle(int title)	title 为标题的资源 id	为 MenuItem 设置标题
setTitle(CharSequence title)	title 为标题的名称	
setTitleCondensed (CharSequence title)	title 为 MenuItem 的缩略标题	设置 MenuItem 的缩略标题，当 MenuItem 不能显示全部的标题时，将显示缩略标题

3) SubMenu 对象

SubMenu 继承自 Menu，每个 SubMenu 实例代表一个子菜单。SubMenu 中常用的方法如表 5-3 所列。

表 5-3　SubMenu 中常用的方法

方 法 名 称	参 数 说 明	方 法 说 明
setHeaderIcon(Drawable icon)	icon 为标题图标 Drawable 对象	设置子菜单的标题图标
setHeaderIcon(int iconRes)	iconRes 为标题图标的资源 id	
setHeaderTitle(int titleRes)	titleRes 为标题文本的资源 id	设置子菜单的标题
setHeaderTitle (CharSequence title)	title 为标题文本对象	
setIcon(Drawable icon)	icon 为图标 Drawable 对象	设置子菜单在父菜单中显示的图标
setIcon(int iconRes)	iconRes 为图标资源 id	
setHeaderView(View view)	view 为用于子菜单标题的 View 对象	设置指定的 View 对象为子菜单图标

2. 菜单选项实例

下面通过两个实例来演示菜单选项的用法。

【例 5-1】 Menu 选项实例。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_1Menu。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个垂直分布的线性布局和一个 ScrollView 控件，在 ScrollView 控件中包含一个 EditText 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/LinearLayout01"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "@drawable/bj1" >
 <! -- 声明一个线性布局 -->
    <ScrollView
        android:id = "@ + id/ScrollView01"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent" >
 <! -- 声明 ScrollView 控件 -->
        <EditText
            android:id = "@ + id/EditText01"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:cursorVisible = "false"
            android:editable = "false"
            android:text = "@string/label" >
 <! -- 声明一个 EditText 控件 -->
        </EditText>
    </ScrollView>
</LinearLayout>
```

(3) 打开 res\values 目录下的 string.xml 文件，在其中声明字符串资源，用作菜单选项的标题以及 EditText 控件的显示内容。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
<string name = "hello"> li5_1Menu </string>
<string name = "app_name"> Menu 实例</string>
 <string name = "action_settings"> Settings </string>
 <string name = "label">您的选择为\n </string>
 <! -- 声明名为 label 的字符串资源 -->
    <string name = "gender">性别</string>
 <! -- 声明名为 gender 的字符串资源 -->
    <string name = "male">男</string>
 <! -- 声明名为 male 的字符串资源 -->
    <string name = "female">女</string>
 <! -- 声明名为 female 的字符串资源 -->
    <string name = "hobby">爱好</string>
 <! -- 声明名为 hobby 的字符串资源 -->
    <string name = "hobby1">游泳</string>
 <! -- 声明名为 hobby1 的字符串资源 -->
```

```
    <string name = "hobby2">唱歌</string>
<! -- 声明名为 hobby2 的字符串资源 -->
    <string name = "hobby3">吃货</string>
<! -- 声明名为 hobby3 的字符串资源 -->
    <string name = "ok">确定</string>
<! -- 声明名为 ok 的字符串资源 -->
</resources>
```

（4）打开 src\fs.li5_1menu 目录下的 MainActivity.java，在文件中重写 onCreate()方法，该方法的主要功能是设置当前的屏幕。初始化菜单的操作通过 onCreateOptionsMenu 方法实现，本程序中的选项菜单共有 3 个选项，分别为性别子菜单、爱好子菜单以及确定子菜单。代码为：

```
public class MainActivity extends Activity {
    final int MENU_GENDER_MALE = 0;                    //性别为男选项编号
    final int MENU_GENDER_FEMALE = 1;                  //性别为女选项编号
    final int MENU_HOBBY1 = 2;                         //爱好 1 选项编号
    final int MENU_HOBBY2 = 3;                         //爱好 2 选项编号
    final int MENU_HOBBY3 = 4;                         //爱好 3 选项编号
    final int MENU_OK = 5;                             //确定菜单选项编号
    final int MENU_GENDER = 6;                         //性别子菜单编号
    final int MENU_HOBBY = 7;                          //爱好子菜单编号

    final int GENDER_GROUP = 0;                        //性别子菜单项组的编号
    final int HOBBY_GROUP = 1;                         //爱好子菜单项组的编号
    final int MAIN_GROUP = 2;                          //外层总菜单项组的编号
    MenuItem[] miaHobby = new MenuItem[3];             //爱好菜单项组
    MenuItem male = null;                              //男性性别菜单项
    @Override
    public void onCreate(Bundle savedInstanceState) {  //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                 //设置当前屏幕
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu){
        //性别单选菜单项组,菜单若编组就是单选菜单项组
          SubMenu subMenuGender = menu.addSubMenu(MAIN_GROUP, MENU_GENDER, 0, R.string.
gender);
        subMenuGender.setIcon(R.drawable.ic_launcher);
        subMenuGender.setHeaderIcon(R.drawable.ic_launcher);
        male = subMenuGender.add(GENDER_GROUP, MENU_GENDER_MALE, 0, R.string.male);
        male.setChecked(true);
        subMenuGender.add(GENDER_GROUP, MENU_GENDER_FEMALE, 0, R.string.female);
        //设置 GENDER_GROUP 组是可选择的、互斥的
        subMenuGender.setGroupCheckable(GENDER_GROUP, true,true);
        //爱好复选菜单项组
        SubMenu subMenuHobby = menu.addSubMenu(MAIN_GROUP,MENU_HOBBY,0,R.string.hobby);
     subMenuHobby.setIcon(R.drawable.hobby);
     miaHobby[0] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY1, 0, R.string.hobby1);
        miaHobby[1] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY2, 0, R.string.hobby2);
        miaHobby[2] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY3, 0, R.string.hobby3);
        miaHobby[0].setCheckable(true);                //设置菜单项为复选菜单项
        miaHobby[1].setCheckable(true);
```

```
        miaHobby[2].setCheckable(true);
        //确定菜单项
        MenuItem ok = menu.add(GENDER_GROUP + 2,MENU_OK,0,R.string.ok);
OnMenuItemClickListener lsn = new OnMenuItemClickListener(){
        //实现菜单项单击事件监听接口
            public boolean onMenuItemClick(MenuItem item) {
                appendStateStr();
                return true;
            }
        };
        ok.setOnMenuItemClickListener(lsn);    //给确定菜单项添加监听器
        //给确定菜单项添加快捷键
        ok.setAlphabeticShortcut('o');         //设置字符快捷键
        //ok.setNumericShortcut('1');          //设置数字快捷键
        //ok.setShortcut('a', '2');            //同时设置两种快捷键
        //注意,同时设置多次时只有最后一个设置起作用
        return true;
    }
    @Override                                  //单选或复选菜单项选中状态变化后的回调方法
    public boolean onOptionsItemSelected(MenuItem mi){
        switch(mi.getItemId()){
          case MENU_GENDER_MALE:               //单选菜单项状态的切换用户要自行写代码完成
          case MENU_GENDER_FEMALE:
                mi.setChecked(true);
                appendStateStr();              //当有效项目变化时记录在文本区中
                break;
          case MENU_HOBBY1:                    //复选菜单项状态的切换要用户自行写代码完成
          case MENU_HOBBY2:
          case MENU_HOBBY3:
              mi.setChecked(!mi.isChecked());
              appendStateStr();                //当有效项目变化时记录在文本区中
              break;
        }
        return true;
    }
                                               //获取当前选择状态的方法
    public void appendStateStr(){
        String result = "您选择的性别为: ";
        if(male.isChecked()){
            result = result + "男";
        }
        else{
            result = result + "女";
        }
        String hobbyStr = "";
        for(MenuItem mi:miaHobby){
            if(mi.isChecked()){
                hobbyStr = hobbyStr + mi.getTitle() + "、";
            }
        }
        if(hobbyStr.length()> 0){
            result = result + ",您的爱好为: " + hobbyStr.substring(0, hobbyStr.length() - 1) + ".\n";
        }
```

```
            else{
                result = result + ".\n";
            }
            EditText et = (EditText)MainActivity.this.findViewById(R.id.EditText01);
            et.append(result);
        }
    }
```

运行程序,效果如图 5-1 所示。当按下右侧的 Menu 键时,效果如图 5-2 所示。





图 5-1　默认界面



图 5-2　显示子菜单界面

如果要实现选项菜单的功能,首先需要重载 OnCreateOptionsMenu()方法创建菜单,然后通过 onOptionsItemSelected()方法对菜单被单击事件进行监听和处理。除了重写 onOptionsItemSelected()方法为菜单单击事件编写响应外,Android 同样允许开发者为不同菜单分别绑定监听器。为菜单项绑定监听器的方法为 setOnMenuItemClickListener。

【例 5-2】　采用简单方法添加菜单项,无须为每个菜单项目指定 id。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_2MenuListener。

(2) 打开 res\layout 目录下的 main.xml 布局文件,设置一个垂直线性布局,在布局文件中声明一个编辑框。代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bj1">
<EditText
```

```
        android:id = "@ + id/txt"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "用于测试的内容"
        android:editable = "false"/>
</LinearLayout >
```

(3) 打开 src\fs. li5_2menulistener 目录下的 MainActivity. java 文件,在文件中通过重写 onOptionsItemSelected(MenuItem mi)方法使处理菜单的单击事件更加简洁。代码为:

```
public class MainActivity extends Activity
{
    private EditText edit;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        edit = (EditText) findViewById(R.id.txt);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        //------------- 向 menu 中添加字体大小的子菜单 -------------
        SubMenu fontMenu = menu.addSubMenu("字体大小");
        //设置菜单的图标
        fontMenu.setIcon(R.drawable.ic_launcher);
        //设置菜单头的图标
        fontMenu.setHeaderIcon(R.drawable.ic_launcher);
        //设置菜单头的标题
        fontMenu.setHeaderTitle("选择字体大小");
        MenuItem font10 = fontMenu.add("10 号字体");
        MenuItem font12 = fontMenu.add("12 号字体");
        MenuItem font14 = fontMenu.add("14 号字体");
        MenuItem font16 = fontMenu.add("16 号字体");
        MenuItem font18 = fontMenu.add("18 号字体");
        //依次为每个菜单项绑定监听器
        font10.setOnMenuItemClickListener(new OnMenuItemClickListener()
        {
            @Override
            public boolean onMenuItemClick(MenuItem item)
            {
                edit.setTextSize(10 * 2);
                return false;
            }
        });
        font12.setOnMenuItemClickListener(new OnMenuItemClickListener()
        {
            @Override
            public boolean onMenuItemClick(MenuItem item)
            {
                edit.setTextSize(12 * 2);
                return false;
            }
```

```
});
font14.setOnMenuItemClickListener(new OnMenuItemClickListener()
{
    @Override
    public boolean onMenuItemClick(MenuItem item)
    {
        edit.setTextSize(14 * 2);
        return false;
    }
});
font16.setOnMenuItemClickListener(new OnMenuItemClickListener()
{
    @Override
    public boolean onMenuItemClick(MenuItem item)
    {
        edit.setTextSize(16 * 2);
        return false;
    }
});
font18.setOnMenuItemClickListener(new OnMenuItemClickListener()
{
    @Override
    public boolean onMenuItemClick(MenuItem item)
    {
        edit.setTextSize(18 * 2);
        return false;
    }
});
//------------- 向 menu 中添加普通菜单项 -------------
MenuItem plain = menu.add("普通菜单项");
plain.setOnMenuItemClickListener(new OnMenuItemClickListener()
{
    @Override
    public boolean onMenuItemClick(MenuItem item)
    {
        Toast toast = Toast.makeText(MainActivity.this
            , "单击了普通菜单项", Toast.LENGTH_SHORT);
        toast.show();
        return false;
    }
});
//------------- 向 menu 中添加文字颜色的子菜单 -------------
SubMenu colorMenu = menu.addSubMenu("字体颜色");
colorMenu.setIcon(R.drawable.color);
//设置菜单头的图标
colorMenu.setHeaderIcon(R.drawable.color);
//设置菜单头的标题
colorMenu.setHeaderTitle("选择文字颜色");
MenuItem redItem = colorMenu.add("红色");
MenuItem greenItem = colorMenu.add("绿色");
MenuItem blueItem = colorMenu.add("蓝色");
//依次为每个菜单项目绑定监听器
redItem.setOnMenuItemClickListener(new OnMenuItemClickListener()
{
```

```
            @Override
            public boolean onMenuItemClick(MenuItem item)
            {
                edit.setTextColor(Color.RED);
                return false;
            }
        });
        greenItem.setOnMenuItemClickListener(new OnMenuItemClickListener()
        {
            @Override
            public boolean onMenuItemClick(MenuItem item)
            {
                edit.setTextColor(Color.GREEN);
                return false;
            }
        });

        blueItem.setOnMenuItemClickListener(new OnMenuItemClickListener()
        {
            @Override
            public boolean onMenuItemClick(MenuItem item)
            {
                edit.setTextColor(Color.BLUE);
                return false;
            }
        });
        return super.onCreateOptionsMenu(menu);
    }
}
```

运行程序,效果如图 5-3 所示。单击子菜单中的“普通菜单项”选项,效果如图 5-4 所示。



图 5-3　所创建的菜单项



图 5-4　为菜单项绑定监听器

5.1.2 上下文菜单属性及实例

1. 上下文菜单属性

上下文菜单(ContextMenu)继承自 Menu。下上文菜单不同于选项菜单,选项菜单服务于 Activity,而上下文菜单是注册到某个 View 对象上的。如果一个 View 对象注册了上下文菜单,则用户可以通过长按(约两秒)该 View 对象弹出上下文菜单。

上下文菜单不支持快捷键(shortcut),其菜单选项也不能带图标,但是可以为上下文菜单的标题指定图标。在使用上下文菜单时经常用到 Activity 类的成员方法,主要方法如下。

- registerForContextMenu(View view):为某个 View 注册菜单。
- onCreateContextMenu(ContextMenu menu,View v,ContextMenuInfo menuInfo):创建 ContextMenu,会在 menu 第一次显示时调用。
- onContextItemSelected(MenuItem item):菜单项被选中后处理选中的菜单项。
- onContextMenuClosed(Menu menu):菜单被关闭的事件。
- openContextMenu(View view):调用打开菜单。
- closeContextMenu():调用关闭菜单。

2. 上下文菜单实例

下面通过一个例子来演示上下文菜单的使用。

【例 5-3】 创建上下文菜单,用于改变编辑框的字体颜色。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_3ContextMenu。

(2) 编写布局文件,布局一个文本框。打开 res\Layout 目录下的 main.xml 文件,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
< TextView
    android:id = "@ + id/txt"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:textSize = "15pt"
    android:text = "可通过上下文菜单修改背景色"/>
</LinearLayout >
```

(3) 编写 Activity 文件,实现上下文菜单。打开 src\com.example.context_m 包下的 MainActivity.java 文件,代码为:

```
public class MainActivity extends Activity
{
```

```
    //菜单项 ID 常量
    private static final int ITEM1 = Menu.FIRST;
    //菜单项 ID 常量
    private static final int ITEM2 = Menu.FIRST + 1;
    //菜单项 ID 常量
    private static final int ITEM3 = Menu.FIRST + 2;
    //声明 TextView 文本视图对象
    private TextView tv;
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //通过 findViewById 方法获得 TextView 实例
        this.tv = (TextView) findViewById(R.id.txt);
        //注册上下文菜单
        this.registerForContextMenu(this.tv);
    }
    public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)
    {
        //添加菜单项
        menu.add(0, ITEM1, 0, "红色背景");
        menu.add(0, ITEM2, 0, "绿色背景");
        menu.add(0, ITEM3, 0, "白色背景");
    }
    public boolean onContextItemSelected(MenuItem item)
    {
        switch (item.getItemId())
        {
            //菜单项 1 被选择
            case ITEM1:
            //设置 TextView 背景色
            this.tv.setBackgroundColor(Color.RED);
            break;
            case ITEM2:
            //设置 TextView 背景色
            this.tv.setBackgroundColor(Color.GREEN);
            break;
            case ITEM3:
            //设置 TextView 背景色
            this.tv.setBackgroundColor(Color.WHITE);
            break;
        }
        return true;
    }
}
```

运行程序,长按文本框,效果如图 5-5 所示。



图 5-5　上下文菜单效果

5.2　点阵图像属性及实例

1. 点阵图像属性

Bitmap 称为点阵图像或绘制图像，是由称作像素(图片元素)的单个点组成的，这些点通过不同的排列和染色构成图样。Bitmap 是 Android 系统中图像处理最重要的类之一，用它可以获取图像文件信息，对图像进行剪切、旋转、缩放等操作，并可以将图像保存成特定格式的文件。Bitmap 位于 android. graphics 包中，Bitmap 不提供对外的构造方法，只能通过 BitmapFactory 类进行实例化。利用 BitmapFactory 的 decodeFile 方法可以方法从特定文件中获取 Bitmap 对象，也可以使用 decodeResource()方法从特定的图像资源中获取 Bitmap 对象。

2. 点阵图像实例

下面通过一个实例来演示 Bitmap 对象的使用。

【例 5-4】 从资源文件中创建 Bitmap 对象，并对其进行一些操作。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_4Bitmap。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中声明一个 SeekBar 组件、一个 ImageView 组件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/LinearLayout01"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
```

```
    android:background = "@drawable/bj1" >
 < SeekBar
       android:id = "@ + id/seekBarId"
       android:layout_width = "fill_parent"
       android:layout_height = "wrap_content"/>
 < ImageView
         android:id = "@ + id/imageView"
         android:layout_width = "wrap_content"
         android:layout_height = "wrap_content"
         android:src = "@drawable/big" />
</LinearLayout >
```

(3) 打开 src\fs. li5_4bitmap 包下的 MainActivity. java 文件,代码为:

```
public class MainActivity extends Activity
{
    ImageView myImageView;
    Bitmap myBmp,newBmp;
    int bmpWidth,bmpHeight;
    SeekBar seekbarRotate;
    float rotAnagle;
    public void onCrate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myImageView = (ImageView)findViewById(R.id.imageView);
        //由 Resource 载入图片
        myBmp = BitmapFactory.decodeResource(getResources(), R.drawable.big);
        bmpWidth = myBmp.getWidth();
        bmpHeight = myBmp.getHeight();
        //实例化 Matrix
        Matrix matrix = new Matrix();
        //设定缩放比例为 1.5
        matrix.postScale(1.5F, 1.5F);
        //顺时针旋转 45°
        matrix.postRotate(45.0F);
        newBmp = Bitmap.createBitmap(myBmp,0,0,bmpWidth,bmpHeight,matrix,true);
        seekbarRotate = (SeekBar)findViewById(R.id.seekBarId);
        seekbarRotate.setOnSeekBarChangeListener(onRotate);
    }
    private SeekBar.OnSeekBarChangeListener onRotate = new SeekBar.OnSeekBarChangeListener()
    {

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
            //TODO 自动生成的方法存根
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
            //TODO 自动生成的方法存根
        }
        @Override
    public void onProgressChanged(SeekBar seekBar, int progress,boolean fromUser) {
            //TODO 自动生成的方法存根
```

```
                Matrix m = new Matrix();
                m.postRotate((float)progress * 3.6F);
                newBmp = Bitmap.createBitmap(myBmp,0,0,bmpWidth,bmpHeight,m,true);
                myImageView.setImageBitmap(newBmp);
            }
        };
    }
```

运行程序，效果如图 5-6 所示。本例实现了拖动进度条图片旋转的效果，使用 BitmapFactory 从资源中载入图片，并获取图片的宽和高，之后使用 Matrix 类对图片进行缩放和旋转操作。



图 5-6　点阵图像效果

5.3　对　话　框

对话框(Dialog)是人机交互操作中十分常见的控件，一般用于在特定条件下对用户显示一些信息，以增强应用的友好性。

Dialog 类是对话框的基类。对话框虽然可以在界面上显示，但是 Dialog 不是 View 类的子类，而是直接继承自 java.lang.Object 类。Dialog 对象也有自己的生命周期，其生命周期由创建它的 Activity 进行管理。Activity 可以调用 showDialog(int id)方法将不同 id 的对话框显示出来，也可以调用 dismissDialog(int id)方法将 id 标识的对话框从用户界面中关闭掉。当 Activity 调用了 showDialog(id)方法，对应 id 的对话框没有被创建时，Android 系统会回调 OnCreateDialog(id)方法创建具有该 id 的对话框。在 Activity 中创建的对话框都会被 Activity 保存，下次 showDialog(id)方法被调用时，如果该 id 对话框已经不创建，则

系统不会再次调用 OnCreateDialog(id)方法创建该对话框,而是回调 onPrepareDialog(int id,Dialog dialog)方法,该方法允许对话框在被显示之前做一些修改。

在 Android 平台主要支持以下几种对话框。

- AlertDialog(提示对话框):AlertDialog 对话框可以包含若干按钮(0～4 个不等)和一些可选的单选按钮或多选按钮等项。一般来说,AlertDialog 的功能能够满足常见的用户界面对话框的需求。
- ProgressDialog(进度对话框):ProgressDialog 可以显示进度轮(wheel)和进度条(bar),由于 ProgressDialog 继承自 AlertDialog,所以进度对话框中也可以添加按钮。
- DatePickerDialog(日期选择对话框):DatePickerDialog 对话框可以显示并允许用户选择日期。
- TimePickerDialog(时间选择对话框):TimePickerDialog 对话框可以显示并允许用户选择时间。

5.3.1 AlertDialog 对话框属性及实例

1. AlertDialog 对话框属性

AlertDialog 对话框是十分常用的用于显示信息的方式,AlertDialog 不能直接通过构造方法构建,而要由 AlertDialog. Builder 类来创建。AlertDialog 对话框的标题、按钮和按钮要响应的事件也由 AlertDialog. Builder 设置。

在使用 AlertDialog. Builder 创建对话框时常用下面几个方法。

- setTitle():设置对话框标题。
- setIcon():设置对话框图标。
- setMessage():设置对话框的提示信息。
- setPositiveButton():为对话框添加确认按钮。
- setNegativeButton():为对话框添加取消按钮。
- setNeutralButton():为对话框添加第 3 个按钮。

2. AlertDialog 对话框实例

下面通过几个实例来演示 AlertDialog 对话框的用法。

【例 5-5】 创建一个简单的 ALertDialog 并显示。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_5AlertDialog1。

(2) 打开 res\layout 目录下的 main. xml 布局文件,代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
```

```
    android:background = "@drawable/bj1">
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

(3) 打开 src\li5_5alertdialog1 目录下的 MainActivity.java 文件，代码为：

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.os.Bundle;
public class MainActivity extends Activity {
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Dialog alertDialog  =  new AlertDialog.Builder(this).
                setTitle("对话框的标题").
                setMessage("对话框的内容").
                setIcon(R.drawable.ic_launcher).
                create();
        alertDialog.show();
    }
}
```

运行程序，效果如图 5-7 所示。



图 5-7　简单的 AlertDiag

例 5-5 很简单，下面在这个 AlertDialog 上添加几个 Button，实现删除操作的提示对话框。

【例 5-6】 创建带按钮的 AlertDialog。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_6AlertDialog2。

(2) 编写 res\layout 目录下的布局文件 main.xml 的代码，与例 5-5 一致。

(3) 打开 src\li5_6alertdialog2 目录下的 MainActivity.java 文件，代码为：

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
public class MainActivity extends Activity {
    /*  第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("确定删除?").
                setMessage("您确定删除该条信息吗?").
                setIcon(R.drawable.ic_launcher).
                setPositiveButton("确定", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动生成的方法存根
                    }
                }).
                setNegativeButton("取消", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动生成的方法存根
                    }
                }).

 setNeutralButton("查看详情", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动生成的方法存根
                    }
                }).
                create();
        alertDialog.show();
    }
}
```

运行程序，效果如图 5-8 所示。

在 Android 中，用 setItems(CharSequence[] items, final OnClickListener listener)方法来实现类似 ListView 的 AlertDialog。第 1 个参数是要显示的数据的数组，第 2 个参数是单击某个 item 的触发事件。

【例 5-7】 创建列表对话框。

其实现步骤如下：



图 5-8　带 3 个按钮的 AlertDialog

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_7AlertDialog3。

(2) 编写 res\layout 目录下的布局文件 main.xml 的代码，与例 5-5 一致。

(3) 打开 src\li5_7alertdialog3 目录下的 MainActivity.java 文件，代码为：

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity {
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final String[] arrayFruit = new String[] { "羽毛球", "乒乓球", "排球", "篮球" };
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("你最喜欢的运动是什么?").
                setIcon(R.drawable.ic_launcher)
                .setItems(arrayFruit, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                   Toast.makeText(MainActivity.this, arrayFruit[which], Toast.LENGTH_
SHORT).show();
                    }
                }).
                setNegativeButton("取消", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动生成的方法存根
```

```
                }
            }).
            create();
        alertDialog.show();
    }
}
```

运行程序，效果如图 5-9 所示。



图 5-9　列表对话框

在 Android 中，用 setSingleChoiceItems(CharSequence[] items, int checkedItem, final OnClickListener listener)方法来实现类似 RadioButton 的 AlertDialog。第 1 个参数是要显示的数据的数组，第 2 个参数是初始值（初始被选中的 item），第 3 个参数是单击某个 item 的触发事件。下面通过一个实例来演示如何创建单选按钮对话框。

【例 5-8】 单选按钮对话框实例。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_8AlertDialog4。

(2) 编写 res\layout 目录下的布局文件 main.xml 的代码，与例 5-5 一致。

(3) 打开 src\li5_8alertdialog4 目录下的 MainActivity.java 文件，代码为：

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity {
    private int selectedFruitIndex = 0;
    /* 第一次调用 Activity 活动 */
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final String[] arrayFruit = new String[] { "羽毛球", "乒乓球", "排球", "篮球" };
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("你最喜欢的运动是什么?").
                setIcon(R.drawable.ic_launcher)
                .setSingleChoiceItems(arrayFruit, 0, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        selectedFruitIndex = which;
                    }
                }).
                setPositiveButton("确认", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(MainActivity.this, arrayFruit[selectedFruitIndex],
Toast.LENGTH_SHORT).show();
                    }
                }).
                setNegativeButton("取消", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动生成的方法存根
                    }
                }).
                create();
        alertDialog.show();
    }
}
```

运行程序,效果如图 5-10 所示。



图 5-10　单选按钮对话框

在 Android 中，用 setMultiChoiceItems（CharSequence[] items，boolean[] checkedItems，final OnMultiChoiceClickListener listener）方法来实现类似 CheckBox 的 AlertDialog。第 1 个参数是要显示的数据的数组，第 2 个参数是选中状态的数组，第 3 个参数是单击某个 item 的触发事件。下面通过一个实例来演示如何创建复选框对话框。

【例 5-9】 创建复选框对话框。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_9AlertDialog5。

(2) 编写 res\layout 目录下的布局文件 main.xml 的代码，与例 5-5 一致。

(3) 打开 src\li5_9alertdialog5 目录下的 MainActivity.java 文件，代码为：

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity {
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final String[] arrayFruit = new String[] { "羽毛球", "乒乓球", "排球", "篮球" };
        final boolean[] arrayFruitSelected = new boolean[] {true, true, false, false};
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("你最喜欢的运动是什么?").
                setIcon(R.drawable.ic_launcher)
                .setMultiChoiceItems(arrayFruit, arrayFruitSelected, new DialogInterface.
OnMultiChoiceClickListener() {
                    @Override
            public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                        arrayFruitSelected[which] = isChecked;
                    }
                }).
                setPositiveButton("确认", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        StringBuilder stringBuilder = new StringBuilder();
                        for (int i = 0; i < arrayFruitSelected.length; i++) {
                            if (arrayFruitSelected[i] == true)
                            {
                                stringBuilder.append(arrayFruit[i] + "、");
                            }
                        }
                        Toast.makeText(MainActivity.this, stringBuilder.toString(), Toast.
LENGTH_SHORT).show();
                    }
                }).

    setNegativeButton("取消", new DialogInterface.OnClickListener() {
                    @Override
```

```
                                    public void onClick(DialogInterface dialog, int which) {
                                        //TODO 自动生成的方法存根
                                    }
                                }).
                                create();
                        alertDialog.show();
                    }
                }
```

运行程序，效果如图 5-11 所示。



图 5-11 复选框对话框

使用 AlertDialog 还可以创建自定义对话框，例如调用 AlertDialog. Builder 的 setAdapter 方法来确定列表项，就可以生成自定义列表项的对话框。下面通过一个实例来创建自定义对话框。

【例 5-10】 定义一个登录对话框。

其实现步骤如下：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li5_10AlertDialog6。

（2）打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal"
    android:background = "@drawable/bj1">
< Button
    android:id = "@ + id/bn"
```

```
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "弹出登录对话框"/>
</LinearLayout>
```

(3) 在 res\layout 目录下创建一个名为 login.xml 的文件，在其中定义 3 个 TextView 控件、3 个 EditText 控件和一个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/loginForm"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1">
<TableRow>
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "用户名:"
    android:textSize = "10pt"/>
<! -- 输入用户名的文本框 -->
<EditText
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:hint = "登录账号"
    android:selectAllOnFocus = "true"/>
</TableRow>
<TableRow>
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "密码:"
    android:textSize = "10pt"/>
<! -- 输入密码的文本框 -->
<EditText
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:password = "true"/>
</TableRow>
<TableRow>
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "电话号码:"
    android:textSize = "10pt"/>
<! -- 输入电话号码的文本框 -->
<EditText
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:hint = "电话号码"
    android:selectAllOnFocus = "true"
    android:phoneNumber = "true"/>
</TableRow>
```

```
<Button
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "注册"/>
</TableLayout>
```

(4) 打开 src\ li5_10alertdialog6 目录下的 MainActivity. java 文件,在该文件中调用 AlertDialog. Builder 的 setView(View view)方法让对话框显示输入界面。代码为:

```
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bn = (Button)findViewById(R.id.bn);
        //定义一个 AlertDialog.Builder 对象
        final Builder builder = new AlertDialog.Builder(this);
        //为按钮绑定事件监听器
        bn.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View source)
            {
                //设置对话框的图标
                builder.setIcon(R.drawable.dcuk2);
                //设置对话框的标题
                builder.setTitle("自定义普通对话框");
                //装载 res\layout\login.xml 界面布局
                TableLayout loginForm = (TableLayout)getLayoutInflater()
                    .inflate( R.layout.login, null);
                //设置对话框显示的 View 对象
                builder.setView(loginForm);
                //为对话框设置一个"登录"按钮
                builder.setPositiveButton("登录"
                    //为按钮设置监听器
                    , new OnClickListener()
                    {
                        @Override
                        public void onClick(DialogInterface dialog, int which)
                        {
                            //此处可执行登录处理
                        }
                    });
                //为对话框设置一个"取消"按钮
                builder.setNegativeButton("取消"
                    ,  new OnClickListener()
                    {
                        @Override
                        public void onClick(DialogInterface dialog, int which)
                        {
                            //取消登录,不做任何事情
                        }
```

```
                    });
                    //创建并显示对话框
                    builder.create().show();
                }
            });
        }
    }
```

运行程序,效果如图 5-12 所示。单击界面中的"弹出登录对话框"按钮,效果如图 5-13 所示。



图 5-12　默认界面



图 5-13　登录界面

还有一种自定义对话框的方式,这种对话框本质上依然是窗口,只是把显示窗口的 Activity 的风格设为对话框风格。

下面通过一个简单实例来演示该对话框的创建。

【例 5-11】　创建一个对话框风格的窗口。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_11AlertDialog7。

(2) 打开 res\layout 目录下的布局文件 main.xml,在文件中定义一个 ImageView 控件和一个 Button 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal"
    android:background = "@drawable/bj1">
< ImageView
```

```
        android:layout_width = "240dp"
        android:layout_height = "wrap_content"
        android:src = "@drawable/dcuk2"      />
    <Button android:id = "@ + id/bn"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "确定"/>
</LinearLayout>
```

(3) 打开 src\li5_11alertdialog7 目录下的 MainActivity.java 文件，为按钮绑定一个监听器，当该按钮被单击时结束该 Activity。代码为：

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bn = (Button)findViewById(R.id.bn);
        //为按钮绑定事件监听器
        bn.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View arg0)
            {
                //结束该 Activity
                finish();
            }
        });
    }
}
```

(4) 打开 AndroidManifest.xml 文件，在其中指定该窗口以对话框风格显示。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.li5_11alertdialog7"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
 <activity
            android:theme = "@android:style/Theme.Dialog"
```

```
            android:name = ".MainActivity"
            android:label = "@string/app_name" >
            <intent - filter >
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
</manifest >
```

运行程序,效果如图 5-14 所示。



图 5-14　对话框风格的窗口

5.3.2　PopupWindow 对话框概述及实例

1. PopupWindow 对话框概述

Android 中的对话框有两种,即 PopupWindow 和 AlertDialog。它们的不同点在于:

(1) AlertDialog 的位置固定,而 PopupWindow 的位置可以随意。

(2) AlertDialog 是非阻塞线程的,而 PopupWindow 是阻塞线程的。

PopupWindow 的位置按照有无偏移可以分为偏移和无偏移两种类型;按照参照物的不同可以分为相对于某个控件(Anchor 锚)和相对于父控件两种类型。

- showAsDropDown(View anchor):相对某个控件的位置(正左下方),无偏移。
- showAsDropDown(View anchor, int xoff, int yoff):相对某个控件的位置,有偏移。
- showAtLocation(View parent, int gravity, int x, int y):相对于父控件的位置(例如正中央 Gravity.CENTER、下方 Gravity.BOTTOM 等),可以设置偏移或无偏移。

2. PopupWindow 对话框实例

下面通过一个例子来说明 PopupWindow 的使用。

【例 5-12】 使用 PopupWindow 创建对话框风格的窗口。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_12PopupWindow。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在该文件中声明 4 个按钮控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "@drawable/bj1">
    < Button
        android:id = "@ + id/button01"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "以自己为 Anchor,不偏移" />
    < Button
        android:id = "@ + id/button02"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "以自己为 Anchor,有偏移" />
    < Button
        android:id = "@ + id/button03"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "以屏幕中心为参照,不偏移(正中间)" />
    < Button
        android:id = "@ + id/button04"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "以屏幕下方为参照,下方中间" />
</LinearLayout >
```

(3) 在 res\layout 目录下创建一个名为 popup_window.xml 的文件，在该文件中声明一个 TextView 控件和一个 RadioGroup 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "#0FFF00"
    android:orientation = "vertical" >
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "选择状态: "
        android:textColor = "@android:color/white"
        android:textSize = "20px" />
    < RadioGroup
```

```
        android:id = "@ + id/radioGroup"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:orientation = "vertical" >
    < RadioButton android:text = "在线" />
    < RadioButton android:text = "离线" />
    < RadioButton android:text = "隐身" />
     </RadioGroup >
</LinearLayout >
```

(4) 打开 src\fs. li5_12popupwindow 目录下的 MainActivity. java 文件,在该文件中实现 PopupWindow 风格窗口的位置偏移效果。代码为:

```
public class MainActivity extends Activity implements OnClickListener,OnCheckedChangeListener
{
    private Button mbutton01;
    private Button mbutton02;
    private Button mbutton03;
    private Button mbutton04;
    private PopupWindow mPopupWindow;
    private int mScreenWidth;                   //屏幕的 width
    private int mScreenHeight;                  //屏幕的 height
    private int mPopupWindowWidth;              //PopupWindow 的 width
    private int mPopupWindowHeight;             //PopupWindow 的 height
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);
        mbutton01  =  (Button) findViewById(R. id. button01);
        mbutton02  =  (Button) findViewById(R. id. button02);
        mbutton03  =  (Button) findViewById(R. id. button03);
        mbutton04  =  (Button) findViewById(R. id. button04);
        mbutton01. setOnClickListener(this);
        mbutton02. setOnClickListener(this);
        mbutton03. setOnClickListener(this);
        mbutton04. setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch (v. getId()) {
        //相对某个控件的位置(正左下方),无偏移
        case R. id. button01:
            getPopupWindowInstance();
            mPopupWindow. showAsDropDown(v);
            break;
        //相对某个控件的位置(正左下方),有偏移
        case R. id. button02:
            getPopupWindowInstance();
            mPopupWindow. showAsDropDown(v, 50, 50);//X,Y 方向各偏移 50
            break;
        //相对于父控件的位置,无偏移
        case R. id. button03:
            getPopupWindowInstance();
            mPopupWindow. showAtLocation(v, Gravity. CENTER, 0, 0);
```

```
                break;
            //相对于父控件的位置,有偏移
            case R.id.button04:
                getPopupWindowInstance();
                mPopupWindow.showAtLocation(v, Gravity.BOTTOM, 0, 50);
                break;
            default:
                break;
            }
        }
        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            mPopupWindow.dismiss();
        }
        //获取 PopupWindow 实例
        private void getPopupWindowInstance() {
            if (null != mPopupWindow) {
                mPopupWindow.dismiss();
                return;
            } else {
                initPopuptWindow();
            }
        }
        //创建 PopupWindow
        private void initPopuptWindow() {
            LayoutInflater layoutInflater = LayoutInflater.from(this);
            View popupWindow = layoutInflater.inflate(R.layout.popup_window, null);
            RadioGroup radioGroup = (RadioGroup) popupWindow.findViewById(R.id.radioGroup);
            radioGroup.setOnCheckedChangeListener(this);
            //创建一个 PopupWindow
            //参数 1: contentView 指定 PopupWindow 的内容
            //参数 2: width 指定 PopupWindow 的 width
            //参数 3: height 指定 PopupWindow 的 height
            mPopupWindow = new PopupWindow(popupWindow, 100, 130);
            //获取屏幕和 PopupWindow 的 width 和 height
            mScreenWidth = getWindowManager().getDefaultDisplay().getWidth();
            mScreenWidth = getWindowManager().getDefaultDisplay().getHeight();
            mPopupWindowWidth = mPopupWindow.getWidth();
            mPopupWindowHeight = mPopupWindow.getHeight();
        }
    }
```

运行程序,单击"以自己为 Anchor,不偏移"按钮,效果如图 5-15 所示。

5.3.3 时间、日期对话框属性及实例

1. 时间、日期对话框概述

在 Android 中使用 DatePickerDialog 实现日期对话框,使用 TimePickerDialog 实现时间对话框,DatePickerDialog 和 TimePickerDialog 的功能比较简单,用法也简单,只要两步即可:

(1) 通过 new 关键字创建 DatePickerDialog、TimePickerDialog 实例,调用它们的 show()方法即可将日期选择对话框、时间选择对话框显示出来。



图 5-15　用 PopupWindow 创建的对话框风格的窗口

（2）为 DatePickerDialog、TimePickerDialog 绑定监听器，这样可以保证用户通过 DatePickerDialog、TimePickerDialog 设置事件时触发监听器，从而通过监听器获取用户设置的事件。

2. 时间、日期对话框实例

下面通过两个例子来演示日期、时间对话框的创建。

【例 5-13】 利用 DatePickerDialog 创建日期对话框。

其实现步骤如下：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li5_13DatePickerDialog。

（2）打开 res\layout 目录下的 main.xml 布局文件，在该文件中声明一个 TextView 控件和一个 Button 控件。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
   android:background="@drawable/bj1">
    <TextView
        android:id="@+id/tv_set_date"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="setDate"
        android:text="设置日期" />
```

```
</LinearLayout>
```

(3) 打开 src\fs. li5_13datepickerdialog 目录下的 MainActivity. java 文件,在该文件中设置对话框监听器。代码为:

```
public class MainActivity extends Activity {
    private static final int DIALOG_DATE_ID = 0;
    //用于显示日期的 TextView
    private TextView tv_set_date;
    //当前系统的年月日
    private int mYear;
    private int mMonth;
    private int mDay;
    /* 第一次调用 Activity */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        tv_set_date = (TextView) findViewById(R.id.tv_set_date);
        Calendar calendar = Calendar.getInstance();
        mYear = calendar.get(Calendar.YEAR);
        mMonth = calendar.get(Calendar.MONTH);
        mDay = calendar.get(Calendar.DAY_OF_MONTH);
        //使用 updateDiaplay()方法把日期显示到 TextView 上
        updateDiaplay();
    }
    private void updateDiaplay(){
        StringBuffer stringBuffer = new StringBuffer();
        stringBuffer.append(mYear).append("年").append(mMonth).append("月").append(mDay).
append("日");
        tv_set_date.setText(stringBuffer);
    }
    //单击按钮调用 setDate()方法
    public void setDate(View v){
        setDate();
    }
    //setDate()方法调用 showDialog(int id)方法,showDialog(int id)方法调用 onCreateDialog
(int id)
    private void setDate() {
        showDialog(DIALOG_DATE_ID);
    }
          @Override
    protected Dialog onCreateDialog(int id) {
        switch(id){
        case DIALOG_DATE_ID:
            //返回一个日期对话框
           return new DatePickerDialog(this, setDateCallBack, mYear, mMonth, mDay);
        }
        return super.onCreateDialog(id);
    }
    //回调函数,int year、int monthOfYear、int dayOfMonth3 个参数为日期对话框设置的日期
    private OnDateSetListener setDateCallBack = new OnDateSetListener() {
        public void onDateSet(DatePicker view, int year, int monthOfYear,
                int dayOfMonth) {
```

```
                mYear = year;
                mMonth = monthOfYear;
                mDay = dayOfMonth;
                updateDiaplay();
            }
        };
    }
```

运行程序,默认界面如图 5-16 所示。单击界面中的“设置日期”按钮,将弹出日期对话框,如图 5-17 所示。



图 5-16 默认界面

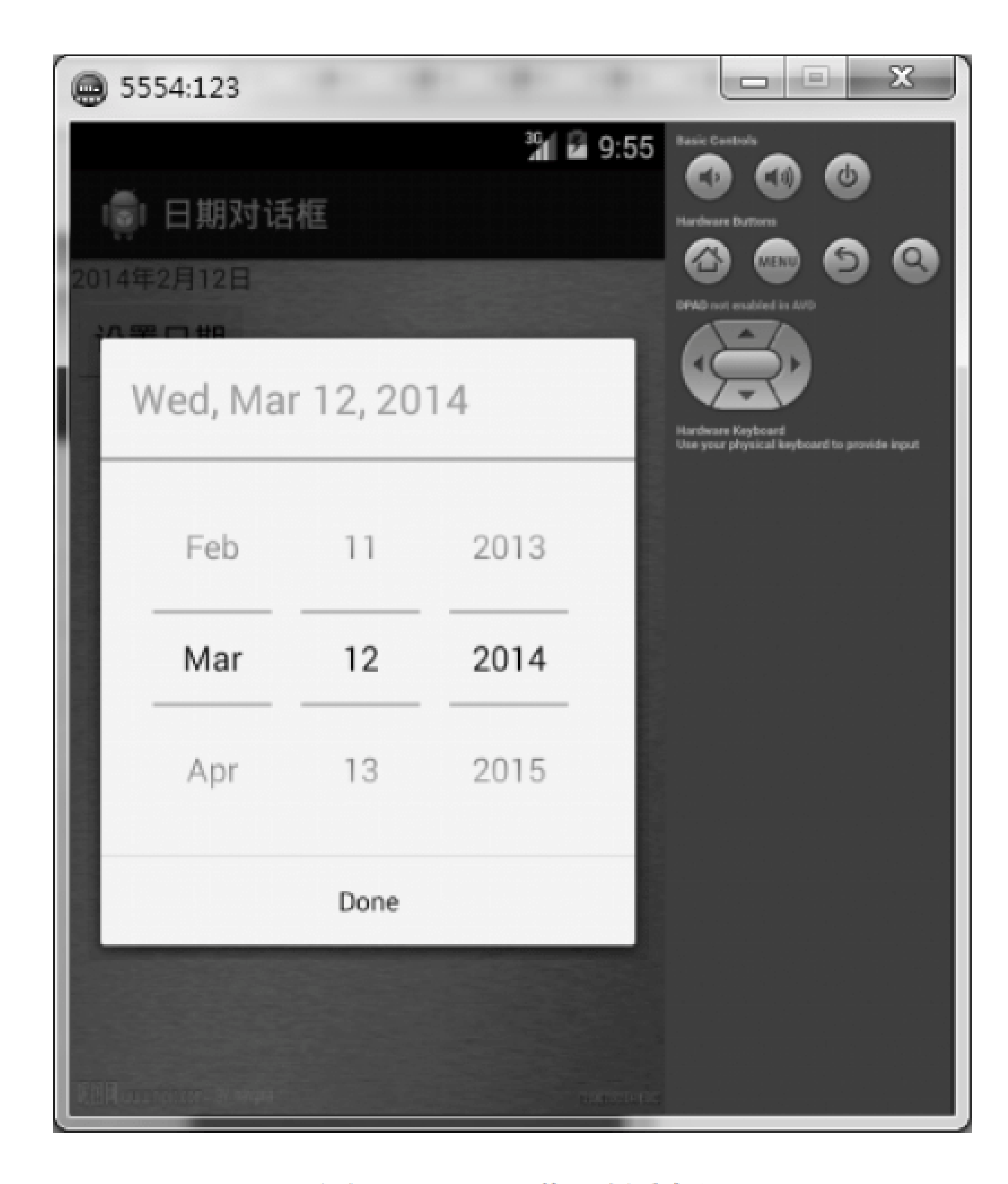


图 5-17 日期对话框

下面使用 DatePickerDialog、TimePickerDialog 创建时间、日期对话框。

【例 5-14】 DatePickerDialog 和 TimePickerDialog 结合使用实例。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_14DateTimeDialog。

(2) 打开 res\layout 目录下的 main. xml 布局文件,在该文件中声明一个 EditText 控件、一个 Button 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:gravity = "center_horizontal"
android:background = "@drawable/bj1">
<EditText
android:id = "@ + id/show"
android:layout_width = "fill_parent"
android:layout_height = "wrap_content"
```

```
android:editable = "false"/>
< Button
android:id = "@ + id/dateBn"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:text = "设置日期与时间"/>
</LinearLayout >
```

(3) 打开 src\fs. li5_14 datetimedialog 目录下的 MainActivity. java 文件，设置日期与时间对话框监听器。代码为：

```
public class MainActivity extends Activity {
    //用来拼接日期和时间,最终用来显示
    StringBuilder str = new StringBuilder("");
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button dateBn = (Button) findViewById(R.id.dateBn);
        //为"设置日期"按钮绑定监听器
        dateBn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View source) {
                Calendar c = Calendar.getInstance();
                //直接创建一个 DatePickerDialog 对话框实例,并将它显示出来
        Dialog dateDialog = new DatePickerDialog(MainActivity.this,          //绑定监听器
                        new DatePickerDialog.OnDateSetListener() {
                    @Override
            public void onDateSet(DatePicker dp, int year,int month, int dayOfMonth) {
                    str.append(year + "-" + (month + 1) + "-" + dayOfMonth + " ");
                        Calendar time = Calendar.getInstance();
        Dialog timeDialog = new TimePickerDialog(MainActivity.this,          //绑定监听器
                            new TimePickerDialog.OnTimeSetListener() {
                        @Override
                    public void onTimeSet(TimePicker tp,int hourOfDay, int minute) {
                            str.append(hourOfDay + ":" + minute);
                    EditText show = (EditText) findViewById(R.id.show);
                            show.setText(str);
                        }
                    }
                    //设置初始时间
                        , time.get(Calendar.HOUR_OF_DAY), time
                        .get(Calendar.MINUTE)
                        //true 表示采用 24 小时制
                        , true);
                        timeDialog.setTitle("请选择时间");
                        timeDialog.show();
                        }
                }
                //设置初始日期
, c.get(Calendar.YEAR), c.get(Calendar.MONTH), c.get(Calendar.DAY_OF_MONTH));
                dateDialog.setTitle("请选择日期");
```

```
                dateDialog.show();
                }
            });
        }
    }
```

运行程序,默认界面如图 5-18(a)所示。单击界面中的"设置日期与时间"按钮,弹出如图 5-18(b)所示的效果;再次单击"设置日期与时间"按钮,弹出如图 5-18(c)所示的效果。



(a) 默认界面



(b) 日期选择界面



(c) 时间选择界面

图 5-18　时间、日期对话框

5.3.4　进度条对话框属性及实例

1. 进度条对话框属性

ProgressDialog 是 AlertDialog 类的一个扩展,可以为一个未定义进度的任务显示一个旋转轮形状的进度动画,或者为一个指定进度的任务显示一个进度条。在一个对话框中显示一个进度条和一个可选的文本信息或一个视图,可以只有文本信息或一个视图,也可以同时使用。

ProgressDialog 的主要实现方法如下。

- setProgressStyle():设置进度条风格。
- setTitle():设置 ProgressDialog 标题。
- setMessage():设置 ProgressDialog 提示信息。
- setIcon():设置 ProgressDialog 标题图标。
- setIndeterminate():设置 ProgressDialog 的进度条是否不明确。
- setCancelable():设置 ProgressDialog 是否可以按退回键取消。
- setButton():设置 ProgressDialog 的一个 Button。
- setProgress():设置 ProgressDialog 进度条的进度。
- show():显示 ProgressDialog。

2. 进度条对话框实例

【例 5-15】 利用 ProgressDialog 创建一个圆形和长形进度条对话框。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_15ProgressDialog。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在该文件中声明两个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1">
 <Button
    android:id = "@ + id/Button01"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "圆形进度条" />
 <Button
    android:id = "@ + id/Button02"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "长形进度条" />
</LinearLayout>
```

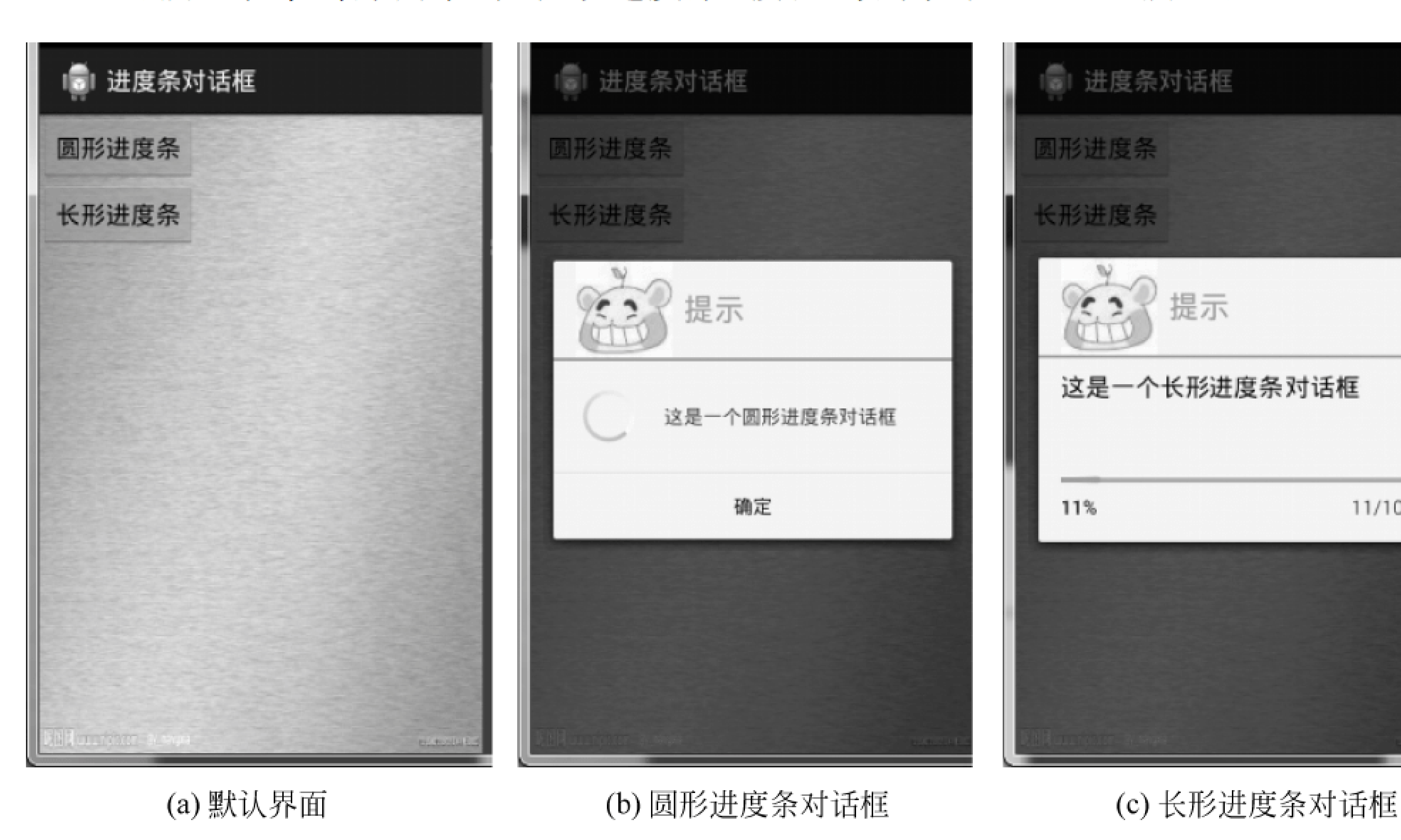
(3) 打开 src\fs. li5_15progressdialog 目录下的 MainActivity. java 文件，在该文件中实现圆形进度条对话框和长形进度条对话框。代码为：

```
public class MainActivity extends Activity
{
    private Button Button1 = null;
    private Button Button2 = null;
    int count = 0;
    //声明进度条对话框
    ProgressDialog progressDialog = null;
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //得到按钮对象
        Button1 = (Button)findViewById(R.id.Button01);
        Button2 = (Button)findViewById(R.id.Button02);
        //设置 mButton01 的事件监听
        Button1.setOnClickListener(new Button1Listener());
        //设置 mButton02 的事件监听
        Button2.setOnClickListener(new Button2Listener());
    }
    private class Button1Listener implements OnClickListener{
        public void onClick(View v) {
            //创建 ProgressDialog 对象
```

```
            progressDialog = new ProgressDialog(MainActivity.this);
            //设置进度条风格,风格为圆形,旋转的
            progressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            //设置 ProgressDialog 标题
            progressDialog.setTitle("提示");
            //设置 ProgressDialog 提示信息
            progressDialog.setMessage("这是一个圆形进度条对话框");
            //设置 ProgressDialog 标题图标
            progressDialog.setIcon(R.drawable.b9);
            //设置 ProgressDialog 的进度条是否不明确
            progressDialog.setIndeterminate(false);
            //设置 ProgressDialog 是否可以按退回键取消
            progressDialog.setCancelable(true);
            //设置 ProgressDialog 的一个 Button
            progressDialog.setButton("确定", new SureButtonListener());
            //让 ProgressDialog 显示
            progressDialog.show();
        }
    }
    //Dialog 中"确定"按钮的监听器
    private class SureButtonListener implements android.content.DialogInterface.OnClickListener{
        public void onClick(DialogInterface dialog, int which) {
            //单击"确定"按钮取消对话框
            dialog.cancel();
        }
    }
    private class Button2Listener implements OnClickListener{
            public void onClick(View v) {
                count = 0;
                //创建 ProgressDialog 对象
                progressDialog = new ProgressDialog(MainActivity.this);
                //设置进度条风格,风格为长形
                progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                //设置 ProgressDialog 标题
                progressDialog.setTitle("提示");
                //设置 ProgressDialog 提示信息
                progressDialog.setMessage("这是一个长形进度条对话框");
                //设置 ProgressDialog 标题图标
                progressDialog.setIcon(R.drawable.b9);
                //设置 ProgressDialog 进度条进度
                progressDialog.setProgress(100);
                //设置 ProgressDialog 的进度条是否不明确
                progressDialog.setIndeterminate(false);
                //设置 ProgressDialog 是否可以按退回键取消
                progressDialog.setCancelable(true);
                //让 ProgressDialog 显示
                progressDialog.show();
                new Thread()
                {
                    public void run()
                    {
                        try
                        {
                            while (count <= 100)
```

```
                        {
                            //由线程来控制进度
                            progressDialog.setProgress(count++);
                            Thread.sleep(100);
                        }
                        progressDialog.cancel();
                    }
                    catch (InterruptedException e)
                    {
                        progressDialog.cancel();
                    }
                }
            }.start();
        }
    }
}
```

运行程序，默认界面如图 5-19(a)所示。单击界面中的“圆形进度条”按钮，效果如图 5-19(b)所示；单击界面中的“长形进度条”按钮，效果如图 5-19(c)所示。



(a) 默认界面　　(b) 圆形进度条对话框　　(c) 长形进度条对话框

图 5-19　进度条对话框

5.4　消息提示框

当程序中有大量消息、图片需要向用户提示时，可考虑使用前面介绍的对话框，但如果程序只有少量信息要向用户呈现，可以考虑使用“便轻量级”的对话框，即 Android 提供的消息提示框。

5.4.1 Toast 概述及实例

1. Toast 概述

Android 中提供了一种简单的 Toast 消息提示框机制，可以在用户单击了某些按钮后给用户提示一些信息，提示的信息不能被用户单击，Toast 的提示信息在经过用户设置的显示时间后会自动消失。通过 Toast 的提示信息可以在调试程序的时候方便地显示某些想显示的东西。

在 Android 中可用下面两种方法创建 Toast。

- makeText(Context context, int resId, int duration)：参数 context 是 toast 显示在哪个上下文，通常是当前 Activity；resId 指显示内容引用 Resource 那条数据，就是从 R 类中去指定显示的消息内容；duration 指定显示时间。Toast 默认有 LENGTH_SHORT 和 LENGTH_LONG 两个常量，分别表示短时间显示和长时间显示。
- makeText(Context context, CharSequence text, int duration)：参数 context 和 duration 与第一个方法相同，对于参数 text，用户可以自己写消息内容。

用上面任意方法创建 Toast 对象之后调用方法 show()即可显示。例如：

```
Toast toast = Toast.makeText(ToastDemoActivity.this, "这是一个普通的 Toast!", Toast.LENGTH_
SHORT);
toast.show();
```

当然，也可以通过以下两种方法设置 Toast 的显示位置。

- setGravity(int gravity, int xOffset, int yOffset)：3 个参数分别表示起点位置、水平向右位移、垂直向下位移。
- setMargin(float horizontalMargin, float verticalMargin)：以横向和纵向的百分比设置显示位置，参数均为 float 类型(水平位移正右负左，竖直位移正上负下)。

例如，设置 Toast 显示位置(起点位置、水平向右位移、垂直向下位移)的 Java 代码为：

```
toast.setGravity(Gravity.TOP | Gravity.LEFT, 0, 200);
```

Toast 显示位置，以横向和纵向的百分比计算，参数均为 float 类型(水平位移正右负左，竖直位移正上负下)，Java 代码为：

```
toast.setMargin( - 0.5f, 0f);
```

Toast 的功能和用法都比较简单，大部分时候它只能显示简单的文本提示。如果应用需要显示诸如图片、列表之类的复杂提示，一般建议使用对话框来完成；如果开发者确实想通过 Toast 来完成，也是可以的，Toast 提供了一个 setView()方法，该方法允许开发者自己定义 Toast 显示的内容。

2. Toast 实例

下面通过一个简单实例来演示 Toast 的用法。

【例 5-16】 创建 5 种 Toast 效果，演示 Toast 的详解。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_16Toast。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在布局文件中声明 5 个按钮控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:padding = "5dip"
    android:gravity = "center"
    android:background = "@drawable/bj1">
< Button android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:id = "@ + id/btnSimpleToast"
    android:text = "默认"/>
< Button android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:text = "自定义显示位置"
    android:id = "@ + id/btnSimpleToastWithCustomPosition"/>
< Button android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:id = "@ + id/btnSimpleToastWithImage"
    android:text = "带图片"/>
< Button android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
android:text = "完全自定义"
android:id = "@ + id/btnCustomToast"/>
< Button android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:text = "其他线程"
    android:id = "@ + id/btnRunToastFromOtherThread"/>
</LinearLayout >
```

(3) 在 res\layout 目录下创建一个名为 custom. xml 的文件，在该文件中声明两个文本框控件和一个图片视图控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
 xmlns:android = "http://schemas.android.com/apk/res/android"
 android:layout_height = "wrap_content" android:layout_width = "wrap_content"
 android:background = "#ffffffff" android:orientation = "vertical"
 android:id = "@ + id/llToast" >
 < TextView
 android:layout_height = "wrap_content"
 android:layout_margin = "1dip"
 android:textColor = "#ffffffff"
 android:layout_width = "fill_parent"
 android:gravity = "center"
 android:background = "#bb000000"
 android:id = "@ + id/tvTitleToast" />
 < LinearLayout
 android:layout_height = "wrap_content"
```

```
android:orientation = "vertical"
android:id = "@ + id/llToastContent"
android:layout_marginLeft = "1dip"
android:layout_marginRight = "1dip"
android:layout_marginBottom = "1dip"
android:layout_width = "wrap_content"
android:padding = "15dip"
android:background = " # 44000000" >
< ImageView
   android:layout_height = "wrap_content"
   android:layout_gravity = "center"
   android:layout_width = "wrap_content"
   android:id = "@ + id/tvImageToast" />
< TextView
   android:layout_height = "wrap_content"
   android:paddingRight = "10dip"
   android:paddingLeft = "10dip"
   android:layout_width = "wrap_content"
   android:gravity = "center"
   android:textColor = " # ff000000"
   android:id = "@ + id/tvTextToast" />
</LinearLayout >
</LinearLayout >
```

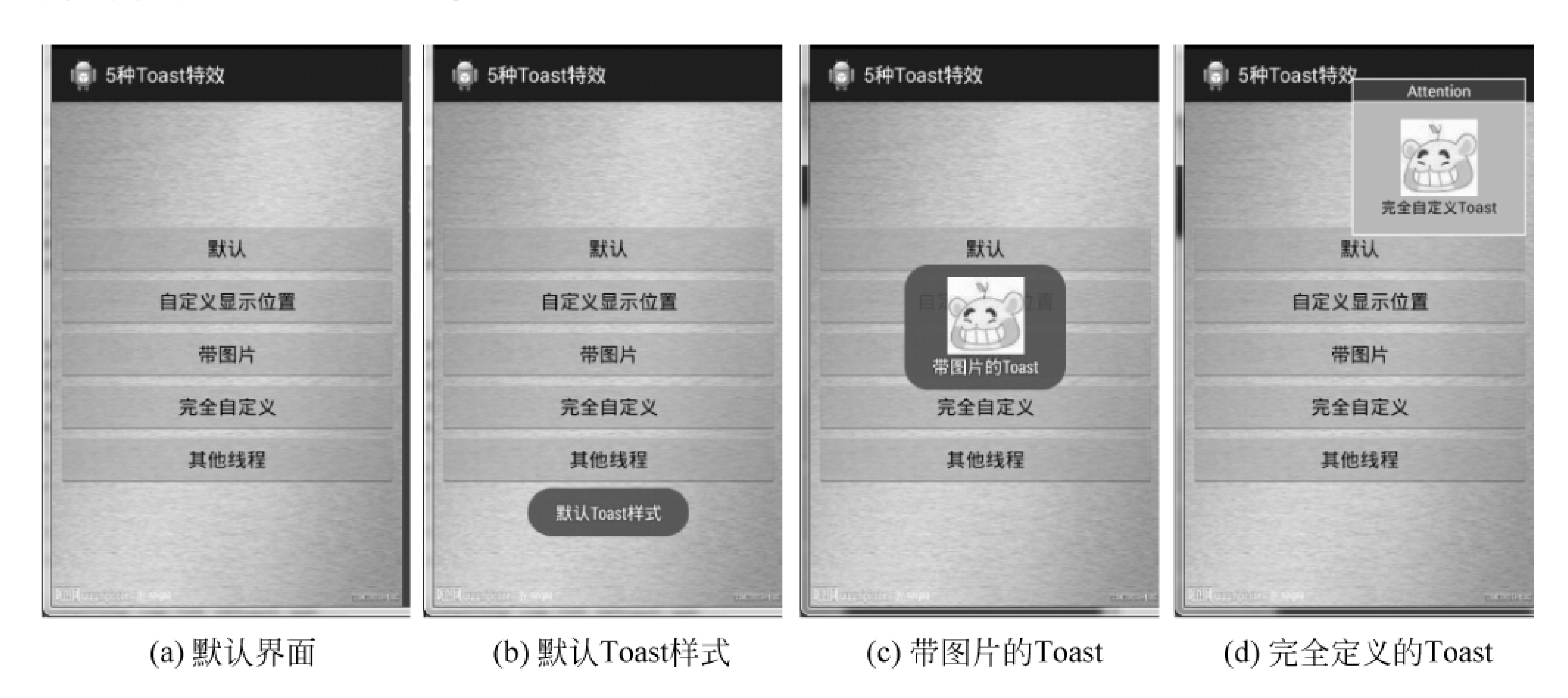
(4) 打开 src\fs. li5_16toast 目录下的 MainActivity. java 文件,创建 5 个 Toast 对象。代码为:

```
public class MainActivity extends Activity implements OnClickListener
{
    Handler handler  =  new Handler();
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        findViewById(R.id.btnSimpleToast).setOnClickListener(this);
        findViewById(R.id.btnSimpleToastWithCustomPosition).setOnClickListener(this);
        findViewById(R.id.btnSimpleToastWithImage).setOnClickListener(this);
        findViewById(R.id.btnCustomToast).setOnClickListener(this);
        findViewById(R.id.btnRunToastFromOtherThread).setOnClickListener(this);
    }
    public void showToast()
    {
        handler.post(new Runnable()
        {
            @Override
            public void run()
            {
                Toast.makeText(getApplicationContext(), "我来自其他线程!",Toast.LENGTH_
SHORT).show();
            }
        });
    }
    @Override
```

```
    public void onClick(View v)
    {
        Toast toast = null;
        switch (v.getId())
        {
            case R.id.btnSimpleToast:
                Toast.makeText(getApplicationContext(), "默认 Toast 样式",
                Toast.LENGTH_SHORT).show();
            break;
            case R.id.btnSimpleToastWithCustomPosition:
                toast = Toast.makeText(getApplicationContext(),
                "自定义位置 Toast", Toast.LENGTH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
            break;
            case R.id.btnSimpleToastWithImage:
                toast = Toast.makeText(getApplicationContext(),
                "带图片的 Toast", Toast.LENGTH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            LinearLayout toastView = (LinearLayout) toast.getView();
            ImageView imageCodeProject = new ImageView(getApplicationContext());
            imageCodeProject.setImageResource(R.drawable.b9);
            toastView.addView(imageCodeProject, 0);
            toast.show();
            break;
            case R.id.btnCustomToast:
                LayoutInflater inflater = getLayoutInflater();
            View layout = inflater.inflate(R.layout.custom,(ViewGroup) findViewById(R.id.
llToast));
            ImageView image = (ImageView) layout.findViewById(R.id.tvImageToast);
            image.setImageResource(R.drawable.b9);
            TextView title = (TextView) layout.findViewById(R.id.tvTitleToast);
            title.setText("Attention");
            TextView text = (TextView) layout.findViewById(R.id.tvTextToast);
            text.setText("完全自定义 Toast");
            toast = new Toast(getApplicationContext());
            toast.setGravity(Gravity.RIGHT | Gravity.TOP, 12, 40);
            toast.setDuration(Toast.LENGTH_LONG);
            toast.setView(layout);
            toast.show();
            break;
            case R.id.btnRunToastFromOtherThread:new Thread(new Runnable()
            {
                public void run()
                {
                    showToast();
                }
            }).start();
        break;
        }
    }
}
```

运行程序，默认界面如图 5-20(a)所示。单击界面中的“默认”按钮，效果如图 5-20(b)

所示；单击界面中的“带图片”按钮，效果如图 5-20(c)所示；单击界面中的“完全自定义”按钮，效果如图 5-20(d)所示。



(a) 默认界面　(b) 默认Toast样式　(c) 带图片的Toast　(d) 完全自定义的Toast

图 5-20　Toast 特效

5.4.2　Notification 概述及实例

1. Notification 概述

在 Android 系统中，应用程序可能会遇到几种情况需要通知用户，有的需要用户回应，有的则不需要，例如：

(1) 当保存文件等事件完成时应该出现一个小的消息，以确认保存成功。

(2) 如果应用程序在后台运行，需要用户注意，应用程序应该创建一个通知，为用户了解他或她的回应提供便利。

(3) 如果应用程序正在执行的工作，用户必须等待(如装载文件)，应用程序应该显示进度或等待提醒。

针对这些情况，Android 提供了不同的提醒方式，主要包括下面几种。

(1) Toast Notification：指出现在屏幕上的暂时性通知，这种通知用于传达一些告知类型的消息，短暂停留后会自动消失，无须用户交互。比如告知下载已完成等。

(2) Status Bar Notification：指以一个图标或者滚动条文本的形式出现在系统顶部状态栏上的通知。当应用程序处于后台运行状态时，这种方式比较合适。这种通知形式的好处是既能被关注到，又无须打断当前任务，可以从顶部下拉查看通知并做选择性处理。

(3) Dialog Notification：类似于 iOS 的 Alert Notification，以对话框窗口的形式出现在屏幕上，用于重要或需及时处理的通知。

2. Notification 实例

在下面的例子中定义了一个 MainActivity 发出广播通知，定义一个 MyReceiver 类继承 Broadcasts 接收通知，在接收完通知之后，启动一个 SecondActivity，在 SecondActivity 类中通过 Notification 和 NotificationManager 可视化显示广播通知。

【例 5-17】 利用 Notification 发出广播通知实例。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_17Notification。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在该文件中声明一个 Button 控件。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/bj1">
 <Button
        android:text="发出广播通知"
        android:id="@+id/Button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

(3) 在 res\layout 目录下创建一个名为 second. xml 的文件，在该文件中声明一个 TextView 控件和一个 Button 控件。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
       android:orientation="vertical"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:background="@drawable/bj1">
<TextView
       android:text="显示通知界面"
       android:id="@+id/TextView1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"/>
<Button
       android:text="取消通知"
       android:id="@+id/cancelButton2"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content" />
</LinearLayout>
```

(4) 打开 src\fs. li5_17notification 目录下的 MainActivity. java 文件，在该文件中为“发出广播通知”按钮绑定监听器。代码为：

```
public class MainActivity extends Activity {
    //声明 Button
    private Button btn;
    //定义 Broadcast Receiver action
    private static final String MY_ACTION = "com.android.notification.MY_ACTION";
    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //设置当前布局视图
        setContentView(R.layout.main);
        //实例化 Button
        btn = (Button)findViewById(R.id.Button1);
        //添加事件监听器
```

```
            btn.setOnClickListener(listener);
            }
        //创建事件监听器
        private OnClickListener listener = new OnClickListener() {
            @Override
            public void onClick(View v) {
                //实例化 Intent
                Intent intent = new Intent();
                //设置 Intent action 属性
                intent.setAction(MY_ACTION);
                //发起广播
                sendBroadcast(intent);
                }
        };
}
```

(5) 在 src\fs.li5_17notification 目录下创建一个名为 Receiver.java 的文件，在该文件中实例化 Intent。代码为：

```
public class Receiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent) {
        //实例化 Intent
        Intent i = new Intent();
        //在新的任务中启动 Activity
        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        //设置 Intent 启动的控件名称
        i.setClass(context, SecondActivity.class);
        //启动 Activity 显示通知
        context.startActivity(i);
        }
}
```

(6) 在 src\fs.li5_17notification 目录下创建一个名为 SecondActivity.java 的文件，在该文件中实现接收通知、为通知添加图标、取消通知等功能。代码为：

```
public class SecondActivity extends Activity {
    //声明按钮
    private Button cancelBtn;
    //声明 Notification
    private Notification notification ;
    //声明 NotificationManager
    private NotificationManager mNotification;
    //Notification 标示 ID
    private static final int ID = 1;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second);
        //实例化按钮
        cancelBtn = (Button)findViewById(R.id.cancelButton2);
        //获得 NotificationManager 实例
        String service = NOTIFICATION_SERVICE;
        mNotification = (NotificationManager)getSystemService(service);
```

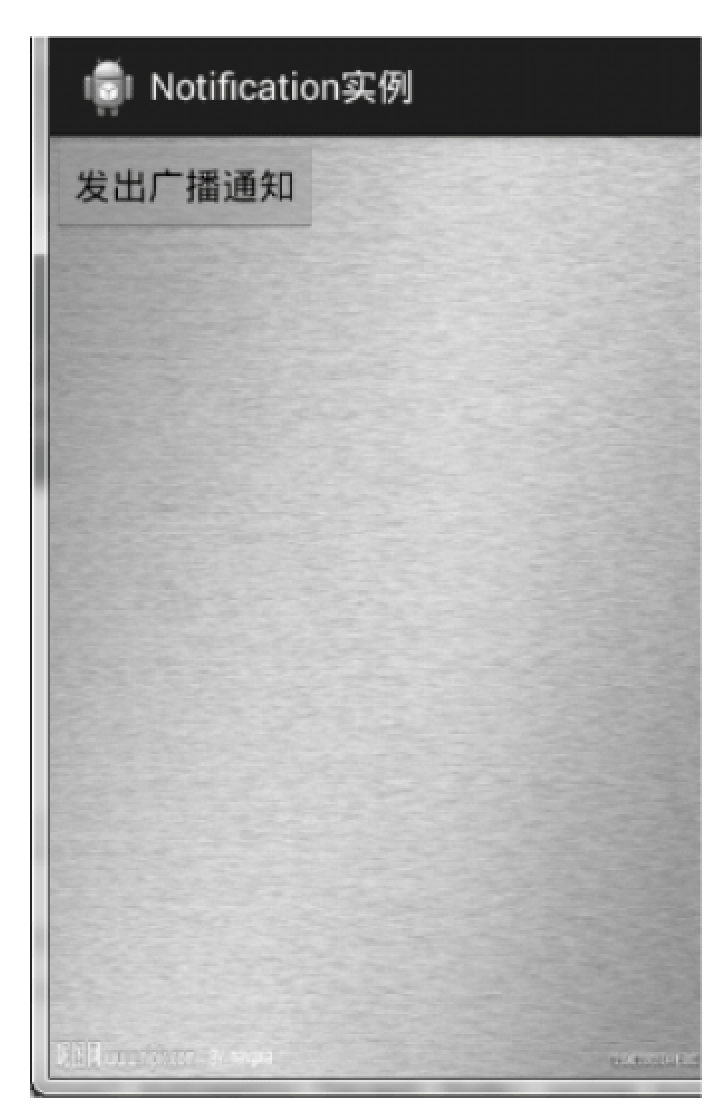
```
        //实例化 Notification
        notification = new Notification();
        //设置显示图标,该图标会在状态栏中显示
        int icon = notification.icon = android.R.drawable.stat_notify_chat;
        //设置显示提示信息,该信息也会在状态栏中显示
        String tickerText = "Test Notification";
        //显示时间
        long when = System.currentTimeMillis();
        notification.icon = icon;
        notification.tickerText = tickerText;
        notification.when = when;
        //实例化 Intent
        Intent intent = new Intent(this, MainActivity.class);
        //获得 PendingIntent
        PendingIntent pi = PendingIntent.getActivity(this, 0, intent, 0);
        //设置事件信息
        notification.setLatestEventInfo(this, "消息", "Hello Android", pi);
        //发出通知
        mNotification.notify(ID, notification);
        //为按钮添加监听器
        cancelBtn.setOnClickListener(cancelListener);
        }
      //取消通知监听器
    private OnClickListener cancelListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            //取消通知
            mNotification.cancel(ID);
        }
    };
}
```

(7) 打开 AndroidManifest.xml 文件,在该文件中加入对 Receiver、SecondActivity 的声明。代码为:

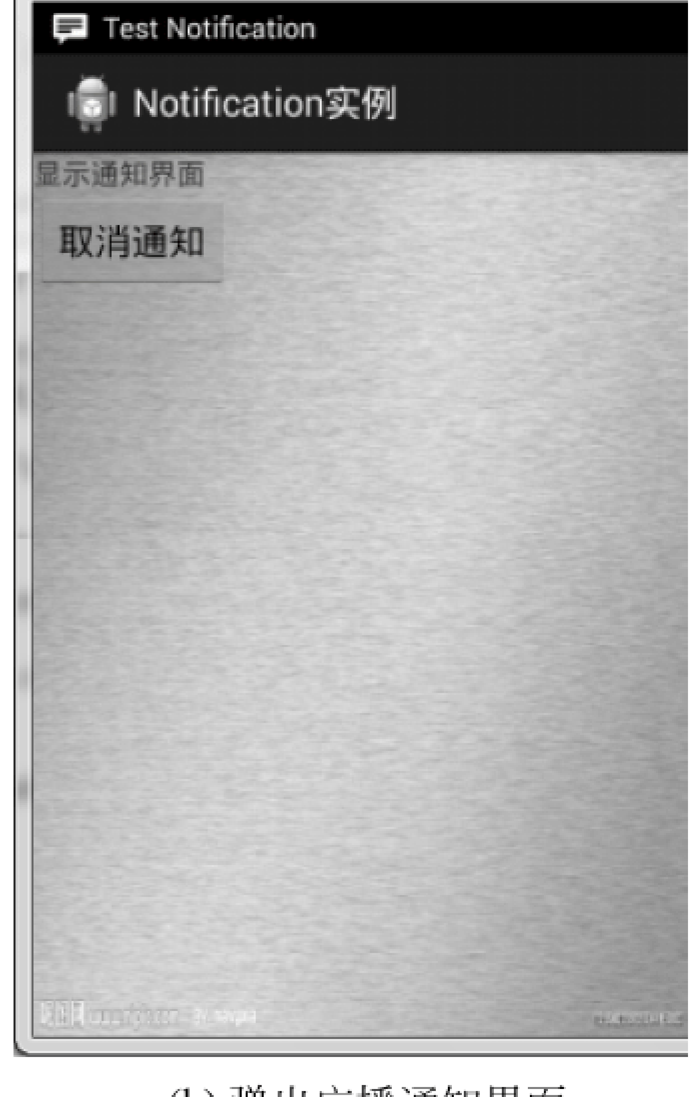
```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.li5_17notification"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.li5_17notification.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
```

```
            </intent - filter>
        </activity>
        <receiver android:name = "Receiver">
        <intent - filter>
            <action android:name = "com.android.notification.MY_ACTION"/>
</intent - filter>
</receiver>
        <activity android:name = "SecondActivity"/>
    </application>
</manifest>
```

运行程序,默认界面如图 5-21(a)所示。单击界面中的“发出广播通知”按钮,效果如图 5-21(b)所示。



(a) 默认界面



(b) 弹出广播通知界面

图 5-21　Notification 广播实例

5.5　菜单与对话框综合实例

本章前面已经对 Android 中的菜单、对话框进行了介绍,本节将通过一个综合实例对这些内容进行综合使用。

【例 5-18】　利用 Android 实现对人的爱好的调查。

爱好调查就是通过用户对问题的回答情况来判断其爱好,需要获取被调查者的姓名和性别、最喜欢的人物以及对人物的评价。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_18Bent。

(2) 将字符串声明到一个文件中,以便于系统的管理与维护。打开 res\values 目录下的 strings.xml 文件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<resources>
    <string name="app_name">li5_18people</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="show_text">爱好调查</string>
    <string name="show_fa">选择最喜欢的人物：</string>
    <string name="show_noreason">输入评价</string>
    <string name="show_sex">性别</string>
    <string name="show_reason">长按给出对人物的评价</string>
    <string name="show_exit">退出</string>
    <string name="show_com">提交</string>
    <string name="show_name">姓名</string>
    <string name="show_noname">输入姓名</string>
</resources>
```

(3) 打开 res\layout 目录下的 main.xml 布局文件，在该文件中声明 6 个 TextView 控件、6 个 ImageView 控件、一个 EditText 控件和一个 Button 控件。代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
     <ImageView
        android:layout_width="60px"
        android:layout_height="60px"
        android:layout_above="@+id/textView1"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="25dp"
        android:src="@drawable/g5" />
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="@string/show_text"/>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/imageView1"
        android:layout_alignParentLeft="true"
        android:text="@string/show_fa" />
    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="120px"
        android:layout_height="120px"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
```

```
        android:layout_marginLeft = "17dp"
        android:src = "@drawable/g4" />
    < ImageView
        android:id = "@ + id/imageView3"
        android:layout_width = "120px"
        android:layout_height = "120px"
        android:layout_centerVertical = "true"
        android:layout_marginLeft = "26dp"
        android:layout_toRightOf = "@ + id/textView1"
        android:src = "@drawable/g5" />
    < ImageView
        android:id = "@ + id/imageView1"
        android:layout_width = "120px"
        android:layout_height = "120px"
        android:layout_alignTop = "@ + id/imageView2"
        android:layout_centerHorizontal = "true"
        android:src = "@drawable/g1" />
    < TextView
        android:id = "@ + id/textView3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/textView1"
        android:layout_marginTop = "20dp"
        android:text = "@string/show_name" />
    < EditText
        android:id = "@ + id/editText"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignBaseline = "@ + id/textView3"
        android:layout_alignBottom = "@ + id/textView3"
        android:layout_marginLeft = "36dp"
        android:layout_toRightOf = "@ + id/textView3"
        android:ems = "10"
        android:text = "@string/show_noname" />
    < TextView
        android:id = "@ + id/textView4"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_above = "@ + id/textView2"
        android:layout_marginBottom = "31dp"
        android:layout_toLeftOf = "@ + id/editText1"
        android:text = "@string/show_sex" />
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignBaseline = "@ + id/textView4"
        android:layout_alignBottom = "@ + id/textView4"
        android:layout_alignLeft = "@ + id/editText1"
        android:layout_alignRight = "@ + id/editText1"
        android:text = "@string/show_sex" />
    < TextView
        android:id = "@ + id/textView5"
```

```
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentLeft = "true"
            android:layout_below = "@ + id/imageView2"
            android:text = "@string/show_reason" />
        < ImageView
            android:id = "@ + id/imageView4"
            android:layout_width = "120px"
            android:layout_height = "120px"
            android:layout_below = "@ + id/textView5"
            android:layout_marginTop = "24dp"
            android:layout_toLeftOf = "@ + id/imageView1"
            android:src = "@drawable/g2" />
        < ImageView
            android:id = "@ + id/imageView5"
            android:layout_width = "120px"
            android:layout_height = "120px"
            android:layout_alignTop = "@ + id/imageView4"
            android:layout_toRightOf = "@ + id/textView1"
            android:src = "@drawable/g3" />
</RelativeLayout >
```

(4) 打开 res\fs. li5_18bent 目录下的 MainActivity. java 文件，代码为：

```
public class MainActivity extends Activity {
    private EditText et_name, et_reason;
    private Button btn_sex;
    private ImageView iview1, iview2, iview3, iview4, iview5;
    private TextView tView, tfaView;
    private String[ ] sexStrings = { "男", "女" };
    private String[ ] fa_name = { "阿狸", "九尾妖狐", "潘斯特" };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        et_name = (EditText) findViewById(R.id.editText);
        btn_sex = (Button) findViewById(R.id.button1);
        iview1 = (ImageView) findViewById(R.id.imageView1);
        iview2 = (ImageView) findViewById(R.id.imageView2);
        iview3 = (ImageView) findViewById(R.id.imageView3);
        iview4 = (ImageView) findViewById(R.id.imageView4);
        iview5 = (ImageView) findViewById(R.id.textView5);
        tfaView = (TextView) findViewById(R.id.textView2);
        tView = (TextView) findViewById(R.id.textView5);
        //选择性别
        btn_sex.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动生成的方法存根
                new AlertDialog.Builder(MainActivity.this)
                        .setTitle("请选择性别")
                        .setIcon(android.R.drawable.ic_dialog_info)
                        .setSingleChoiceItems(sexStrings, 0,
                        new DialogInterface.OnClickListener() {
```

```
                public void onClick(DialogInterface dialog,int which) {
btn_sex.setText(sexStrings[which]);
dialog.dismiss();
                                    }
                              }).setNegativeButton("取消", null).show();
                }
        });
        //选择喜欢的人物
        OnClickListener clickListener = new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动生成的方法存根
                switch (v.getId()) {
                case R.id.imageView1:
                    tfaView.setText("下列人物中,你最喜欢的是阿狸");
                    break;
                case R.id.imageView2:
                tfaView.setText("下列人物中,你最喜欢的是九尾妖狐");
                    break;
                case R.id.imageView3:
                    tfaView.setText("下列人物中,你最喜欢的是潘斯特");
                    break;
                default:
                    break;
                }
            }
        };
        iview1.setOnClickListener(clickListener);
        iview2.setOnClickListener(clickListener);
        iview3.setOnClickListener(clickListener);
OnLongClickListener longClickListener = new OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
                //TODO 自动生成的方法存根
                et_reason = new EditText(MainActivity.this);
                new AlertDialog.Builder(MainActivity.this)
                        .setTitle("输入理由")
                        .setIcon(android.R.drawable.ic_dialog_info)
                        .setView(et_reason)
                        .setPositiveButton("确定",
                                new DialogInterface.OnClickListener() {
                                    @Override
public void onClick(DialogInterface dialog,int which) {
                        String reasonString = et_reason
                            .getText().toString();
                                f (reasonString.equals("")) {
                                    reasonString = "输入评价";
                                    }
                                    tView.setText(reasonString);
                                    }
                            }).setNegativeButton("取消", null).show();
                return false;
            }
        };
```

```
            iview4.setOnLongClickListener(longClickListener);
            iview5.setOnLongClickListener(longClickListener);
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            return true;
        }
    }
```

第 6 章　Android 图形与动画

6.1　Android　图　形

Android 系统提供了华丽的界面效果，对于开发者来说，同样可以使用丰富的绘图资源使自己的应用程序更加丰富多彩。在 Android 系统中，主要通过 Graphics 软件包来绘制图形。Graphics 软件包中提供了 Canvas(画布)、Paint(画笔)等常用的类，通过这些类中的方法，可以方便地绘制点、线、颜色以及各种几何图形等。

6.1.1　画笔

在 Android 中，绘图操作一般是通过 Paint 画笔在 Canvas 画布上进行绘制的，最后将 Canvas 画布呈现给用户。在绘图之前首先需要设置 Paint 画笔，在 Android 中通过 Paint 类来实现，Paint 类中提供了很多方法来设置画笔属性。

1. 图形绘制

Paint 类实现图形绘制的主要方法如下。

- setARGB(int a,int r,int g,int b)：设置绘制的颜色，a 代表透明度，r、g、b 代表颜色值。
- setAlpha(int a)：设置绘制图形的透明度。
- setColor(int color)：设置绘制的颜色，使用颜色值来表示，该颜色值包括透明度和 RGB 颜色。
- setAntiAlias(boolean aa)：设置是否使用抗锯齿功能，如果使用会消耗较多资源，绘制图形的速度会变慢。
- setDither(boolean dither)：设定是否使用图像抖动处理，如果使用会使绘制出来的图片颜色更加平滑和饱满，图像更加清晰。
- setFilterBitmap(boolean filter)：如果设置为 true，则图像在动画过程中会滤掉对 Bitmap 图像的优化操作，加快显示速度，本设置项依赖于 dither 和 xfermode 的设置。
- setMaskFilter(MaskFilter maskfilter)：设置 MaskFilter，可以用不同的 MaskFilter 实现滤镜的效果，如滤化、立体等。
- setColorFilter(ColorFilter colorfilter)：设置颜色过滤器，可以在绘制颜色时实现不用颜色的变换效果。
- setPathEffect(PathEffect effect)：设置绘制路径的效果，如点画线等。

- setShader(Shader shader)：设置图像效果，使用 Shader 可以绘制出各种渐变效果。
- setShadowLayer(float radius ,float dx,float dy,int color)：在图形下面设置阴影层，产生阴影效果。radius 为阴影的角度，dx 和 dy 为阴影在 X 轴和 Y 轴上的距离，color 为阴影的颜色。
- setStyle(Paint. Style style)：设置画笔的样式，取值为 FILL、FILL_OR_STROKE 或 STROKE。
- setStrokeCap(Paint. Cap cap)：当画笔样式为 STROKE 或 FILL_OR_STROKE 时，设置笔刷的图形样式，如圆形样式(Cap. ROUND)或方形样式(Cap. SQUARE)。
- setSrokeJoin(Paint. Join join)：设置绘制时各图形的结合方式，如平滑效果等。
- setStrokeWidth(float width)：当画笔样式为 STROKE 或 FILL_OR_STROKE 时，设置笔刷的粗细度。
- setXfermode(Xfermode xfermode)：设置图形重叠时的处理方式，如合并，取交集或并集，经常用来制作橡皮的擦除效果。

2. 文本绘制

Paint 类实现文本绘制的主要方法如下。

- setFakeBoldText(boolean fakeBoldText)：模拟实现粗体文字，设置在小字体上效果会非常差。
- setSubpixelText(boolean subpixelText)：设置该项为 true，将有助于文本在 LCD 屏幕上的显示效果。
- setTextAlign(Paint. Align align)：设置绘制文字的对齐方向。
- setTextScaleX(float scaleX)：设置绘制文字在 X 轴上的缩放比例，可以实现文字的拉伸效果。
- setTextSize(float textSize)：设置绘制文字的字号大小。
- setTextSkewX(float skewX)：设置斜体文字，skewX 为倾斜弧度。
- setTypeface(Typeface typeface)：设置 Typeface 对象，即字体风格，包括粗体、斜体以及衬线体、非衬线体等。
- setUnderlineText(boolean underlineText)：设置带有下划线的文字效果。
- setStrikeThruText(boolean strikeThruText)：设置带有删除线的效果。

3. 画笔实例

下面给出对应的两个实例，分别实现画笔的使用。

【例 6-1】 实现画笔的颜色设置。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li6_1PaintSet1。

(2) 打开 res\layout 目录下的布局文件 main. xml，代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
```

```
        android:paddingLeft = "@dimen/activity_horizontal_margin"
        android:paddingRight = "@dimen/activity_horizontal_margin"
        android:paddingTop = "@dimen/activity_vertical_margin"
        tools:context = ".MainActivity" >
</RelativeLayout>
```

(3) 打开 src\fs.li6_1paintset1 包下的 MainActivity.java 文件，定义及调用自定义的 MyPaint 类。代码为：

```
public class MainActivity extends Activity {
    private MyPaint myPaint = null;              //声明自定义 View 对象
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {   //重载 onCreate 方法
        super.onCreate(savedInstanceState);
        this.myPaint = new MyPaint(this);        //创建自定义 View 对象
        setContentView(myPaint);                 //设置显示自定义 View
    }
}
```

(4) 在 src\fs.li6_1paintset1 包下新建一个名为 MyPaint.java 的类，该类继承于 View 类，并构建了 Paint 对象。在此重载了 onDraw 方法，在其中使用 setColor 方法来设置画笔为绿色，接着使用该画笔在 Canavs 画布上绘制了直线和矩形。完成设置后，还需要在 Activity 中设置显示这个自定义的 View，通过 setContentView 方法来实现。代码为：

```
public class MyPaint extends View implements Runnable
{   //自定义 View
    private Paint paint = null;                         //声明画笔对象
    public MyPaint (Context context)
    {
        super(context);
        //TODO 自动存根法
        paint = new Paint();                            //构建对象
        new Thread(this).start();                       //开启线程
    }
    @Override
    protected void onDraw(Canvas canvas)
    {   //重载 onDraw 方法
        super.onDraw(canvas);
        paint.setColor(Color.BLUE);                     //设置画笔颜色
        canvas.drawColor(Color.WHITE);
        canvas.drawLine(50, 50, 450, 50, paint);        //绘制直线
        canvas.drawRect(100, 100, 200, 600, paint);     //绘制矩形
        canvas.drawRect(300, 100, 400, 600, paint);     //绘制矩形
    }
    @Override
    public void run() {                                 //重载 run 方法
        while(!Thread.currentThread().isInterrupted())
        {
            try
            {
                Thread.sleep(100);
```

```
                }
                catch(InterruptedException e)
                {
                    Thread.currentThread().interrupt();
                }
                postInvalidate();                          //更新界面
            }
        }
}
```

运行程序，效果如图 6-1 所示。



图 6-1　设置画笔的颜色

【例 6-2】　设置画笔的透明度。

其实现步骤如下：

（1）在 Eclipse 中创建 Android 应用项目，命名为 li6_2PaintAlpha。

（2）打开 res\layout 目录下的布局文件 main.xml，代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
</RelativeLayout>
```

(3) 打开 src\fs. li6_2paintalpha 包下的 MainActivity. java 文件，声明和调用自定义的 MyPaintAlpha 类。代码为：

```
public class MainActivity extends Activity
{
    private MyPaintAlpha myPaintAlpha = null;           //声明自定义 View 对象
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {   //重载 onCreate 方法
        super.onCreate(savedInstanceState);
        this.myPaintAlpha = new MyPaintAlpha(this);     //创建自定义 View 对象
        setContentView(myPaintAlpha);                   //设置显示自定义 View
    }
}
```

(4) 在 src\fs. li6_2paintalpha 包下新建一个名为 MyPaintAlpha. java 的类，该类用于设置画笔的颜色，然后设置透明度为 50，接着用此画笔绘制直线和矩形。代码为：

```
public class MyPaintAlpha extends View implements Runnable
{   //自定义 View
    private Paint paint = null;                         //声明画笔对象
    public MyPaintAlpha(Context context)
    {
        super(context);
        //TODO 自动存根法
        paint = new Paint();                            //构建对象
        new Thread(this).start();                       //开启线程
    }
    protected void onDraw(Canvas canvas)
    {   //重载 onDraw 方法
        super.onDraw(canvas);
        paint.setColor(Color.BLUE);                     //设置画笔颜色
        paint.setAlpha(50);                             //设置画笔的透明度
        canvas.drawColor(Color.WHITE);
        canvas.drawLine(50, 50, 450, 50, paint);        //绘制直线
        canvas.drawRect(100, 100, 200, 600, paint);     //绘制矩形
        canvas.drawRect(300, 100, 400, 600, paint);     //绘制矩形
    }
    @Override
    public void run()
    {                                                   //重载 run 方法
        //TODO 自动生成的方法存根
        while(!Thread.currentThread().isInterrupted())
        {
            try
            {
                Thread.sleep(100);
            }
            catch(InterruptedException e)
            {
                Thread.currentThread().interrupt();
```

```
                }
                postInvalidate();                          //更新界面
            }
        }
    }
```

运行程序,效果如图 6-2 所示。



图 6-2　设置笔画的透明度

6.1.2　画布

Android 系统中的绘图操作主要是在 Canvas 画布上进行的,在绘图时,使用的是前面设置好的 Paint 画笔。在 Android 系统中,Canvas 类提供了很多常用的图形,例如直线、矩形、圆形、文字等,同时,也可以为画布设置颜色、尺寸等。Canvas 画布是主要的绘图场所。

1. 画布方法

Canvas 提供了以下一些方法。

1) drawColor 方法

用于设置画布的背景颜色,可以通过 Color 类中的预定义颜色来设置,也可以通过指定 RGB 值来设置。其语法格式为:

```
Public void drawColor(int color)
```

其中,参数 color 为颜色值。用户也可以直接使用系统 Color 类中定义的颜色。

2) drawLine 方法

用于在画布上绘制直线,通过指定直线的两个端点坐标来绘制。该方法只能绘制单条直线,如果需要同时绘制多条直线,可以使用 drawLines 方法。其语法格式为:

```
Public void drawLine(float startX,float startY,float stopX,float,stopY,Paint paint)
```

其中,参数 startX 为起始端点的 X 坐标;参数 startY 为起始端点的 Y 坐标;参数 stopX 为终止端点的 X 坐标;参数 stopY 为终止端点的 Y 坐标;参数 paint 为绘制直线所使用的画笔。

3) drawLines 方法

用于在画布上绘制多条直线,通过指定直线的端点坐标数组来绘制。该方法可以绘制多条直线,非常灵活。其语法格式为:

```
Public void drawLines(float[] pts,Paint paint)
```

其中,参数 pts 为绘制直线的端点数组,每条直线占用 4 个数据;参数 paint 为绘制直线所使用的画笔。

4) drawPoint 方法

用于在画布上绘制一个点,通过指定端点坐标来绘制。该方法只能绘制单个点,如果需要同时绘制多个点,可以使用 drawPoints 方法。其语法格式为:

```
Public void drawPoint(float x,float y,Paint paint)
```

其中,参数 x 为绘制点的 X 坐标;参数 y 为绘制点的 Y 坐标;参数 paint 为绘制点所使用的画笔。

5) drawPoints 方法

用于在画布上绘制多个点,通过指定端点坐标数组来绘制。该方法可以绘制多个点,同时可以指定哪些点绘制,哪些点不绘制,非常灵活。其语法格式为:

```
Public void drawPoints(float[] pts,Paint paint)
Public void drawPoints(float[] pts,int offset,int count,Paint paint)
```

其中,参数 pts 为绘制点的数组,每个端点占用两个数据;参数 offset 为跳过的数据个数,这些数据将不参与绘制过程;参数 count 为实际参与绘制的数据个数;参数 paint 为绘制时所使用的画笔。

6) drawRect 方法

用于在画布上绘制矩形,可以通过指定矩形的 4 条边来实现,也可以通过指定 Rect 对象来实现,同时可以通过设置画笔的空心效果来绘制空心的矩形。其语法格式为:

```
Public void drawRect(Rect r,Paint paint);
Public void drawRect(RectF rect,Paint paint);
Public void drawRect(float left,float top,float right,float below,Paint paint)
```

其中,参数 r 为 Rect 对象;参数 rect 为 RectF 对象;参数 left 为矩形的左边位置;参数 top 为矩形的上边位置;参数 right 为行矩形的右边位置;参数 below 为矩形的下边位置;参数 paint 为绘制矩形时所使用的画笔。

7) drawRoundRect 方法

用于在画面上绘制圆角矩形,通过指定 RectF 对象和圆角半径来实现。该方法是绘制圆角矩形的主要方法,同时也可以通过设置画笔的空心效果来绘制空心的圆角矩形。其语法格式为:

```
Public void drawRoundRect(RectF rect,float rx,float ry,Paint paint)
```

其中,参数 rect 为 RectF 对象;参数 rx 为 X 方向上的圆角半径;参数 ry 为 Y 方向上的圆角半径;参数 paint 为绘制时所使用的画笔。

8) drawCircle 方法

用于在画布上绘制圆形,通过指定圆形圆心的坐标和半径来实现。该方法是绘制圆形的主要方法,同时也可以通过设置画笔的空心效果来绘制空心的圆形。其语法格式为:

```
Public void drawCircle(float cx,float cy,float radius,Paint paint)
```

其中,参数 cx 为圆心的 X 坐标;参数 cy 为圆心的 Y 坐标;参数 radius 为圆的半径;参数 paint 为绘制时所使用的画笔。

9) drawOval 方法

用于在画布上绘制椭圆形,通过指定椭圆外切矩形的 RectF 对象来实现。该方法为绘制椭圆形的主要方法,同时也可以通过设置画笔的空心效果来绘制空心的椭圆形。其语法格式为:

```
Public void drawOval(RectF oval,Paint paint)
```

其中,参数 oval 为椭圆外切矩形的 RectF 对象;参数 paint 为绘制时所使用的画笔。

10) drawPath 方法

用于在画布上绘制任意多边形,通过指定 Path 对象来实现。在 Path 对象中规划了多边形的路径信息。其语法格式为:

```
Public void drawPath(Path path,Paint paint)
```

其中,参数 path 为包含路径信息的 Path 对象;参数 paint 为绘制时所使用的画笔。

11) drawArc 方法

用于在画布上绘制圆弧,通过指定圆弧所在的椭圆对象、起始角度、终止角度来实现。该方法是绘制圆弧的主要方法,其语法格式为:

```
Public void drawArc(RectF oval, float startAngle, float sweepAngle, Boolean useCenter, Paint
paint)
```

其中,参数 oval 为圆弧所在的椭圆对象;参数 startAngle 为圆弧的起始角度;参数 sweepAngle 为圆弧的角度;参数 useCenter 为是否显示半径连线,当取值为 true 时表示显示圆弧与圆心的半径连线,当取值为 false 时表示不显示;参数 paint 为绘制时所使用的画笔。

12) drawText 方法

用于在画布上绘制字符串,通过指定字符串的内容和显示的位置来实现。在画布上绘制字符串是经常用到的操作,Android 系统提供了非常灵活的绘制字符串的方法,可以根据不同的需要调用不同的方法来实现。字体的大小、样式等信息都需要在 Paint 画笔中指定。其语法格式为:

```
Public void drawText(String text,float x,float y,Paint paint)
Public void drawText(char[] text,int index,int count,float x,float y,Paint paint)
```

```
Public void drawText(CharSequence text, int start, int end, float x, float y, Paint paint)
Public void drawText(String text, int start, int end, float x, float y, Paint paint)
```

其中,参数 text 为字符串内容,可以采用 String 格式,也可以采用 char 字符数组形式;参数 x 为显示位置的 X 坐标;参数 y 为显示位置的 Y 坐标;参数 index 为显示的起始字符位置;参数 count 为显示字符的个数;参数 start 为显示的起始字符位置;参数 end 为显示的终止字符位置;参数 paint 为绘制时所使用的画笔。

13) drawBitmap 方法

用于在画布上绘制图像,通过指定 Bitmap 对象来实现。前面的各个方法都是自己绘制各个图形,但我们的应用程序往往需要直接引用一些图片资源,这时即可使用 drawBitmap 方法在画布上直接显示图像。其语法格式为:

```
Public void drawBitmap(Bitmap bitmap, float left, float top, Paint paint)
```

其中,参数 bitmap 为 Bitmap 对象,代表了图像资源;参数 left 为图像显示的左边位置;参数 top 为图像显示的上边位置;参数 paint 为绘制时所使用的画笔。

14) save 方法

用于锁定画布,这种方法主要用于锁定画布中的某一个或几个对象,对锁定对象操作的场合。使用 save 方法锁定画布并完成操作之后,需要使用 restore 方法解除锁定。其语法格式为:

```
Public int save()
```

15) restore 方法

用于解除锁定的画布,这种方法主要用在 save 方法之后,使用 save 方法锁定画布并完成操作后,需要使用 restore 方法解除锁定。

16) clipRect 方法

用于裁剪画布,即设置画布的显示区域。在使用时,可以使用 Rect 对象来指定裁剪区,也可以通过指定矩形的 4 条边来指定裁剪区。该方法主要用于部分显示以及对画布中的部分对象进行操作的场合。其语法格式为:

```
Public Boolean clipRect(Rect rect)
Public Boolean clipRect(float left, float top, float right, float bottom)
Public Boolean clipRect(int left, int top, int right, int bottom)
```

其中,参数 rect 为 Rect 对象,用于定义裁剪区的范围;参数 left 为矩形裁剪区的左边位置,可以为浮点型或整型;参数 top 为矩形裁剪区的上边位置,可以为浮点型或整型;参数 right 为矩形裁剪区的右边位置,可以为浮点型或整型;参数 bottom 为矩形裁剪区的下边位置,可以为浮点型或整型。

17) rotate 方法

用于旋转画布,通过旋转画布,可以将画布上绘制的对象旋转。在使用这个方法时,将会把画布上的所有对象都旋转。为了只对某一个对象进行旋转,可以通过 save 方法锁定画布,然后执行旋转操作,最后通过 restore 方法解锁,此后再绘制其他图形。其语法格式为:

```
Public void rotate(float degrees)
Public final void rotate(float degrees,float px,float py)
```

其中,参数 degrees 为旋转角度,正数为顺时针方向,负数为逆时针方向;参数 px 为旋转点的 X 坐标;参数 py 为旋转点的 Y 坐标。

2. 画布实例

下面通过一个实例来演示画布的使用。

在实际游戏开发中,可能需要对某个精灵执行旋转、缩放和一些其他操作。可以通过旋转画布来实现,但是旋转画布时会旋转画布上的所有对象,而只是需要旋转其中的一个,这时就需要用到 save 方法来锁定需要操作的对象,在操作之后再通过 restore 方法来解除锁定。

【例 6-3】 演示怎样在 Android 中绘制基本的集合图形。

该程序中关键在于一个自定义的 View 组件,其重写了 onDraw(Canvas)方法,接下来在该 Canvas 上绘制大量的几何图形。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 li6_3Canvase。

(2) 打开 res\layout 目录下的布局文件 main. xml,文件代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
< fs.li6_3canvase.CanvasView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 打开 res\values 目录下的 strings. xml 文件,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
   < string name = "app_name"> Canvas 与 Paint </string >
   < string name = "hello_world"> Hello_world!</string >
   < string name = "action_settings"> Settings </string >
   < string name = "circle">圆形</string >
   < string name = "square">正方形</string >
   < string name = "rect">长方形</string >
   < string name = "round_rect">圆角矩形</string >
   < string name = "oval">椭圆形</string >
   < string name = "triangle">三角形</string >
   < string name = "pentagon">五角形</string >
</resources >
```

(4) 打开 src\fs. li6_3canvase 包下的 MainActivity. java 文件,代码为:

```
package fs.li6_3canvase;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity
```

```
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

(5) 在 src\fs. li6_3canvase 包下新建一个名为 CanvasView. java 的文件，在该文件中的 Canvas 画布中绘制各种图形。代码为：

```
package fs.li6_3canvase;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.LinearGradient;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.RectF;
import android.graphics.Shader;
import android.graphics.drawable.BitmapDrawable;
import android.util.AttributeSet;
import android.view.View;
public class CanvasView extends View
{
    public CanvasView(Context context, AttributeSet set)
    {
        super(context, set);
    }
    @Override
    //重写该方法,进行绘图
    protected void onDraw(Canvas canvas)
    {
        super.onDraw(canvas);
        //把整张画布绘制成白色
        canvas.drawColor(Color.WHITE);
        Paint paint = new Paint();
        //去锯齿
        paint.setAntiAlias(true);
        paint.setColor(Color.BLUE);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(3);
        //绘制圆形
        canvas.drawCircle(40, 40, 30, paint);
        //绘制正方形
        canvas.drawRect(10, 80, 70, 140, paint);
        //绘制矩形
        canvas.drawRect(10, 150, 70, 190, paint);
        RectF re1 = new RectF(10, 200, 70, 230);
        //绘制圆角矩形
        canvas.drawRoundRect(re1, 15, 15, paint);
        RectF re11 = new RectF(10, 240, 70, 270);
```

```
//绘制椭圆
canvas.drawOval(re11, paint);
//定义一个 Path 对象,封闭成一个三角形
Path path1 = new Path();
path1.moveTo(10, 340);
path1.lineTo(70, 340);
path1.lineTo(40, 290);
path1.close();
//根据 Path 进行绘制,绘制三角形
canvas.drawPath(path1, paint);
//定义一个 Path 对象,封闭成一个五角形
Path path2 = new Path();
path2.moveTo(26, 360);
path2.lineTo(54, 360);
path2.lineTo(70, 392);
path2.lineTo(40, 420);
path2.lineTo(10, 392);
path2.close();
//根据 Path 进行绘制,绘制五角形
canvas.drawPath(path2, paint);
//设置填充风格后绘制
paint.setStyle(Paint.Style.FILL);
paint.setColor(Color.RED);
canvas.drawCircle(120, 40, 30, paint);
//绘制正方形
canvas.drawRect(90, 80, 150, 140, paint);
//绘制矩形
canvas.drawRect(90, 150, 150, 190, paint);
RectF re2 = new RectF(90, 200, 150, 230);
//绘制圆角矩形
canvas.drawRoundRect(re2, 15, 15, paint);
RectF re21 = new RectF(90, 240, 150, 270);
//绘制椭圆
canvas.drawOval(re21, paint);
Path path3 = new Path();
path3.moveTo(90, 340);
path3.lineTo(150, 340);
path3.lineTo(120, 290);
path3.close();
//绘制三角形
canvas.drawPath(path3, paint);
Path path4 = new Path();
path4.moveTo(106, 360);
path4.lineTo(134, 360);
path4.lineTo(150, 392);
path4.lineTo(120, 420);
path4.lineTo(90, 392);
path4.close();
//绘制五角形
canvas.drawPath(path4, paint);
//设置渐变器后绘制
//为 Paint 设置渐变器
Shader mShader = new LinearGradient(0, 0, 40, 60
    , new int[] {
```

```
            Color.RED, Color.GREEN, Color.BLUE, Color.YELLOW }
            , null , Shader.TileMode.REPEAT);
        paint.setShader(mShader);
        //设置阴影
        paint.setShadowLayer(45 , 10 , 10 , Color.GRAY);
        //绘制圆形
        canvas.drawCircle(200, 40, 30, paint);
        //绘制正方形
        canvas.drawRect(170, 80, 230, 140, paint);
        //绘制矩形
        canvas.drawRect(170, 150, 230, 190, paint);
        RectF re3 = new RectF(170, 200, 230, 230);
        //绘制圆角矩形
        canvas.drawRoundRect(re3, 15, 15, paint);
        RectF re31 = new RectF(170, 240, 230, 270);
        //绘制椭圆
        canvas.drawOval(re31, paint);
        Path path5 = new Path();
        path5.moveTo(170, 340);
        path5.lineTo(230, 340);
        path5.lineTo(200, 290);
        path5.close();
        //根据 Path 进行绘制,绘制三角形
        canvas.drawPath(path5, paint);
        Path path6 = new Path();
        path6.moveTo(186, 360);
        path6.lineTo(214, 360);
        path6.lineTo(230, 392);
        path6.lineTo(200, 420);
        path6.lineTo(170, 392);
        path6.close();
        //根据 Path 进行绘制,绘制五角形
        canvas.drawPath(path6, paint);
        //设置字符大小后绘制
        paint.setTextSize(24);
        paint.setShader(null);
        Bitmap bitmap = null;
        //绘制 7 个字符串
        canvas.drawText(getResources().getString(R.string.circle), 240, 50,paint);
        canvas.drawText(getResources().getString(R.string.square), 240, 120,paint);
        canvas.drawText(getResources().getString(R.string.rect), 240, 175,paint);
        canvas.drawText(getResources().getString(R.string.round_rect), 230,220, paint);
        canvas.drawText(getResources().getString(R.string.oval), 240,260, paint);
        canvas.drawText(getResources().getString(R.string.triangle), 240, 325,paint);
        canvas.drawText(getResources().getString(R.string.pentagon), 240, 390,paint);
        //显示图形
            bitmap = ((BitmapDrawable)getResources().getDrawable(R.drawable.ra)).getBitmap();
        canvas.drawBitmap(bitmap, 50, 450, null);       //绘制图像
    }
}
```

运行程序,效果如图 6-3 所示。

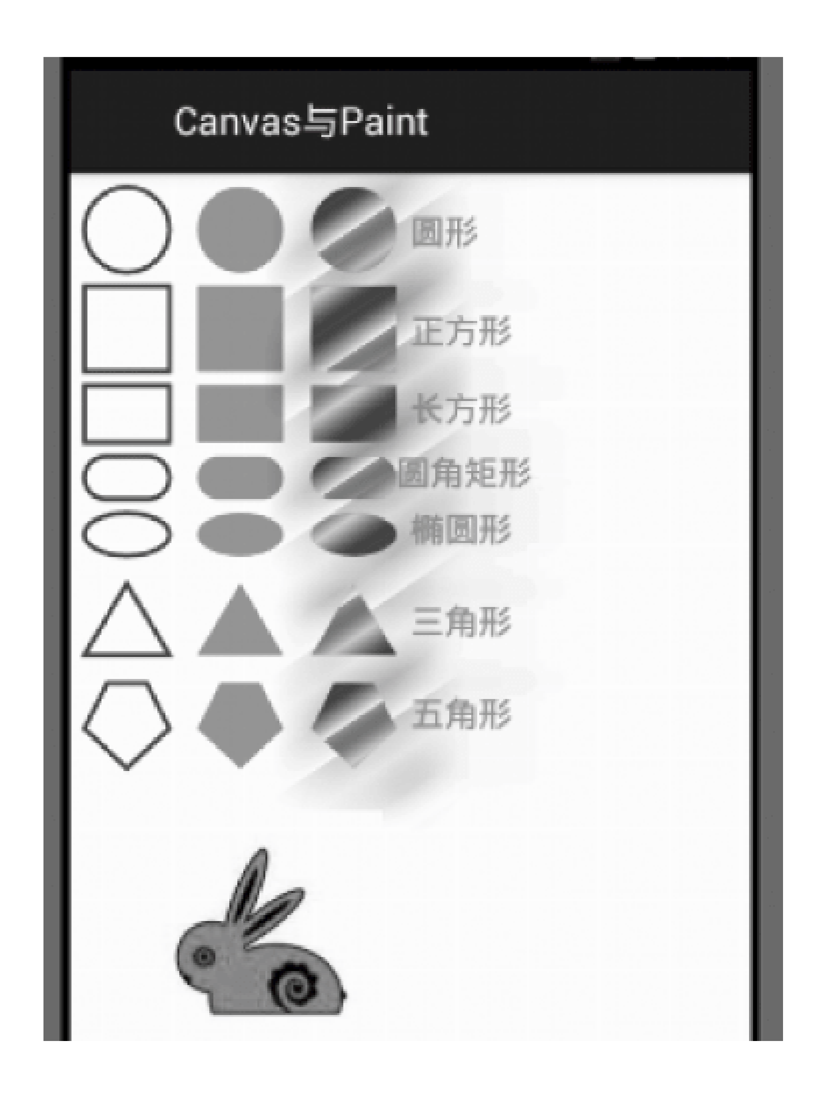


图 6-3　Canvas 与 Paint 绘图效果

6.2　Path 绘图

Android 提供的 Path 是一个非常有用的类，其可以预先在 View 上将 *N* 个点连成一条"路径"，然后调用 Canvas 的 drawPath(path, paint)即可沿着路径绘制图形。实际上，Android 还为路径绘制提供了 PathEffect 来定义绘制效果，PathEffect 包含以下子类(每个子类代表一种绘制效果)：

- ComposePathEffect
- CornerPathEffect
- DashPathEffect
- DiscretePathEffect
- PathDashPathEffect
- SumPathEffect

下面通过一个实例让读者了解这些绘制效果。

【例 6-4】　Path 绘图实例。

以下代码绘制 7 条路径，分别演示了不使用效果和使用上面 6 种子类效果。

```
public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this));
    }
    class MyView extends View
    {
        float phase;
```

```
PathEffect[] effects = new PathEffect[7];
int[] colors;
private Paint paint;
Path path;
public MyView(Context context)
{
    super(context);
    paint = new Paint();
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeWidth(4);
    //创建并初始化 Path
    path = new Path();
    path.moveTo(0, 0);
    for (int i = 1; i <= 15; i++)
    {
        //生成 15 个点,随机生成它们的 Y 坐标,并将它们连成一条 Path
        path.lineTo(i * 30, (float) Math.random() * 80);
    }
    //初始化 7 个颜色
    colors = new int[] {Color.BLACK, Color.BLUE, Color.CYAN
        , Color.GREEN, Color.MAGENTA, Color.RED ,Color.BLUE};
}
@Override
protected void onDraw(Canvas canvas)
{
    //将背景填充成白色
    canvas.drawColor(Color.WHITE);
    /*
    * 下面开始初始化 7 种路径效果
    */
    //不使用路径效果
    effects[0] = null;
    //使用 CornerPathEffect 路径效果
    effects[1] = new CornerPathEffect(10);
    //初始化 DiscretePathEffect
    effects[2] = new DiscretePathEffect(3.0f ,5.0f);
    //初始化 DashPathEffect
    effects[3] = new DashPathEffect(new float[]
        { 20, 10, 5, 10 }, phase);
    //初始化 PathDashPathEffect
    Path p = new Path();
    p.addRect(0 , 0, 8, 8, Path.Direction.CCW);
    effects[4] = new PathDashPathEffect(p, 12, phase,
        PathDashPathEffect.Style.ROTATE);
    //初始化 PathDashPathEffect
    effects[5] = new ComposePathEffect(effects[2], effects[4]);
    effects[6] = new SumPathEffect(effects[4], effects[3]);
    //将画布移动到(8,8)处开始绘制
    canvas.translate(8, 8);
    //依次使用 7 种不同路径效果、7 种不同的颜色来绘制路径
    for (int i = 0; i < effects.length; i++)
```

```
                {
                    paint.setPathEffect(effects[i]);
                    paint.setColor(colors[i]);
                    canvas.drawPath(path, paint);
                    canvas.translate(0, 60);
                }
                //改变 phase 值,形成动画效果
                phase += 1;
                invalidate();
            }
        }
    }
```

运行程序,效果如图 6-4 所示。

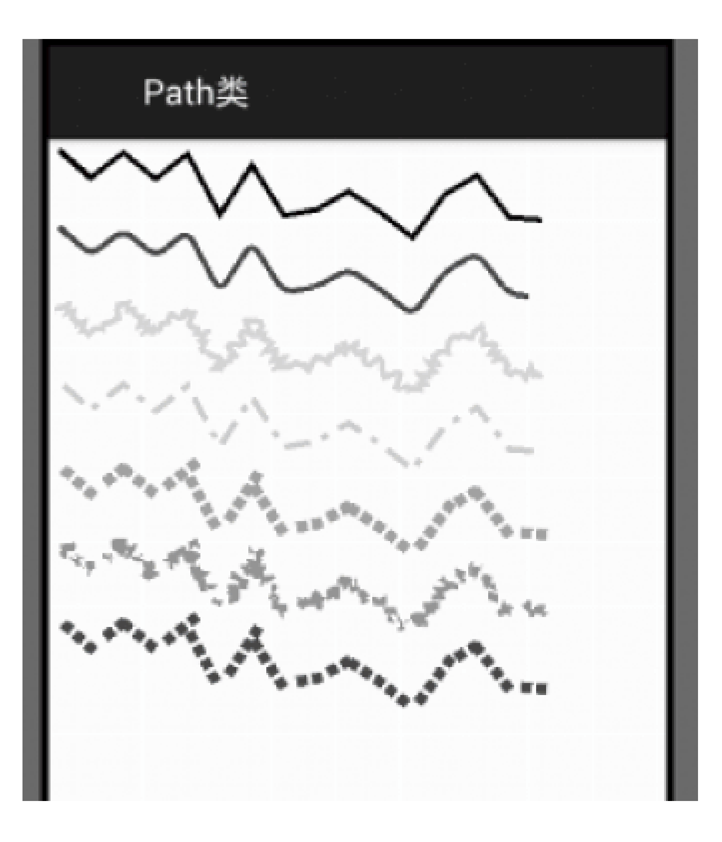


图 6-4　设置 Path 路径效果

6.3　美化 UI 控件

本节主要掌握以下技术:

- 使用 style 在不同事件下控制控件外观;
- 美化常用控件的方法。

6.3.1　使用 style

在 Android 系统中,样式定义在"\android-sdk-windows\platforms\data\res\values"文件夹的 styles.xml 文件中,这里面有系统所有的样式定义声明,但有一些样式是隐藏的,它们使用@hide 作为标记,例如下面的样式代码:

```
<! - @hide -->
<style name="TextAppearance.SearchResult.Title">
  <item name="android:textSize">16sp</item>
</style>
```

样式 TextAppearance.SearchResult.Title 在 ADT 的自动提示中是不显示的,因为它是隐藏的。

使用系统自带的样式非常简单,在名称为 com.li6_4style 的项目中的 main.xml 代码为:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/textView1"
        style="@android:style/TextAppearance.Large"
```

```
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "大字显示 Style 样式" />
    < TextView
        style = "@android:style/TextAppearance.Small"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/textView1"
        android:layout_below = "@ + id/textView1"
        android:layout_marginTop = "70dp"
        android:text = "小字显示 Style 样式" />
</RelativeLayout >
```

上面的代码使用的是系统自带的样式，显示的外观是大字体和小字体，效果如图 6-5 所示。



图 6-5　大字体与小字体

值得注意的是，在 ADT 中将自动提示的样式名称中的“_”(下划线)改成小数点“.”即可。

style 样式为系统中的资源，在 Android 中使用资源要注意以下几个知识点。

(1) 引用自定义资源：@资源类型/资源名称

这种写法是使用用户自定义的资源名称，例如下面的代码：

```
< TextView android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/helllo"
    android:background = "@color/ghyColor"/>
```

通过使用@string 和@color 即可引用对应资源类型的自定义资源名称。

(2) 引用系统资源与使用隐藏资源：@android 资源类型/资源名称

sdk 文件夹“android-sdk-windows\platforms\data\res\value”中的 colors. xml 配置文件有系统默认的 color 颜色配置，在项目中可以引用系统资源。代码为：

```
< TextView android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/hello"
    android:background = "@ * android:color/hint_foreground_dark"/>
```

代码使用了 color. xml 文件中的 hint_foreground_dark 样式 style，但由于此样式在 public. xml 中并未定义，所以在项目中并不能直接使用，这时使用@ * android 的方式引用隐藏的资源，加入“ * ”(星号)的作用是使用系统隐藏的资源，即使用非 public 的资源。在 Android 项目中可以使用的资源在路径“android-sdk-windows\platforms\data\res\value”的 public. xml 文件中。在这里需要说明一下，没有在 public. xml 中声明的资源是 Google 不推荐使用的。

1. style 概述

定义 style 样式资源可以把 UI 用户界面进行美化及改良，样式可以应用于一个或更多控件，也可以应用于一个或更多 Activity 对象，还可以应用于整个应用程序。

使用 style 样式非常简单，在 Style_2 项目中的 res\value\文件夹下创建一个名为 style.xml 的文件，样式的文件名任意，但是为了文件名有意义，应尽量给文件名加入 style 的关键字，从而快速识别 XML 文件资源的类型，内容为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <style name = "ghyStyle1">
    </style>
</resources>
```

在这个样式中并没有对样式添加任何定义属性，即在<style>标签中并没有<item>标签，但<style>标签的属性 name 却代表了这个样式的名称 ghyStyle1，这个名称也在 R.java 文件中进行了注册，即样式资源的 id，代码为：

```
public static final class style
{
public static final int ghyStyle1 = 0x7f050000;
}
```

虽然定义了一个名称为 ghyStyle1 的样式，但是却没有细节的定义，继续更改 style.xml 中的样式代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <style name = "ghyStyle1">
     <item name = "android:textSize">40dip<\item>
    </style>
</resources>
```

上面代码中的<item>标签的 name 属性值 android:textSize 来自于 android-sdk-windows\platforms\data\res\value 文件夹中的 attrs.xml 文件，在此文件中定义了 Android 系统自带的所有属性，其中就有 textSize 属性的声明，代码为：

```
</ul>
   -->
<attr name = "textSize" format = "dimension"/>
```

这段样式定义文字的大小为 40dip，<item>标签定义样式的细节信息，name 属性定义样式的名称，而<item>的 body 体定义样式的值。

更加详细的 style 语法为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <style name = "style_name "parent = "@[package:]style/style_to_inherit">
     <item name = "[package:]style_property_name">style_value</item>
    </style>
</resources>
```

在定义样式时有以下几点值得注意：

- 样式的存放路径是在 res\value 文件夹下。
- 元素<resources>是必须具有的标签，是 XML 样式文件的根(root)结点。
- 元素<style>：style 标签是定义一个样式，它有名称为<item>的子结点。<style>的 name 属性可以生成此样式的资源 id，在 R.java 文件中，通过这个 resourceId 即可将这个样式应用到 View 控件、Activity 或整个应用程序中。<style>还有 parent 属性，这个属性定义当前的样式从哪个样式继承下来，使得样式也可以得到代码的重用。
- 元素<item>定义样式的属性，是<style>标签的标签，具有 name 属性，用于定义样式属性的具体名称。

虽然定义了样式，那么怎样引用呢？其语法格式为：

```
@[package:]style/style_name
```

2. style 的继承与使用

在 Style_2 项目中，将 main.xml 中的<textView>控件应用前面创建的 ghyStyle1 样式，布局代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    <TextView
        style = "@style/ghyStyle1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

运行程序，效果如图 6-6 所示。

下面来实现一个样式的继承，继续更改 style.xml 的样式，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <style name = "ghyParentStyle">
        <item name = "android:background"> # FF000F </item>
    </style>
    <style name = "ghyStyle1" parent = "@style/ghyParentStyle">
        <item name = "android:textSize"> 40dip </item>
    </style>
</resources>
```

在 style.xml 文件中定义一个名称为 ghyParentStyle 的父样式，然后在 ghyStyle1 中进行继承，运行程序，效果如图 6-7 所示。

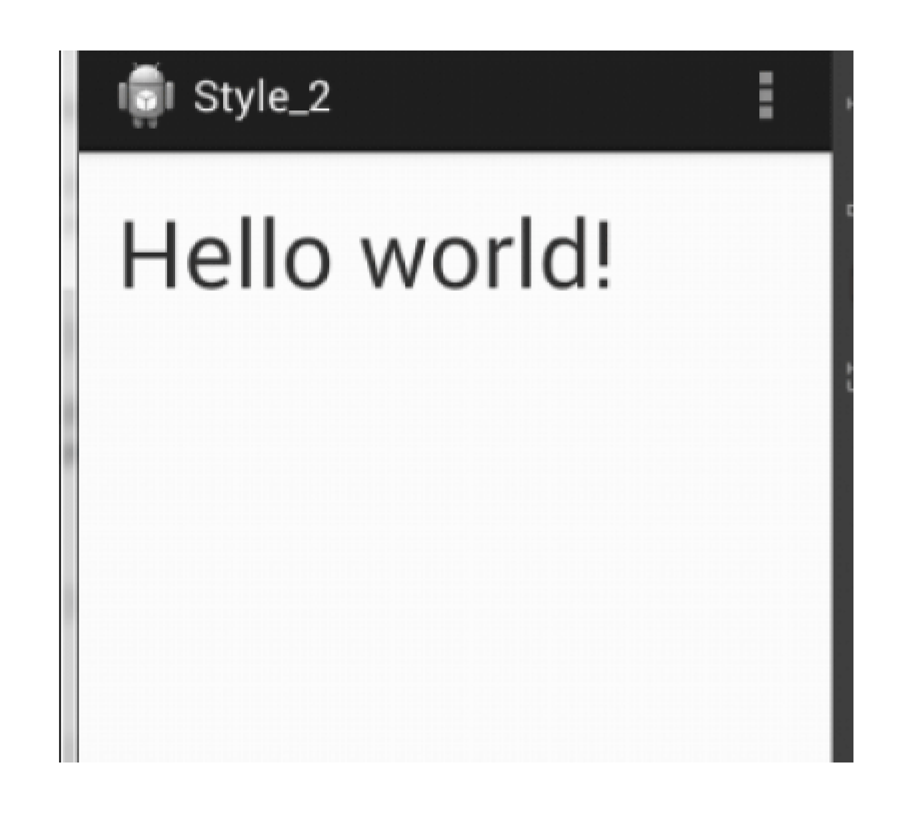



图 6-6　TextView 控件应用 ghyStyle1 样式

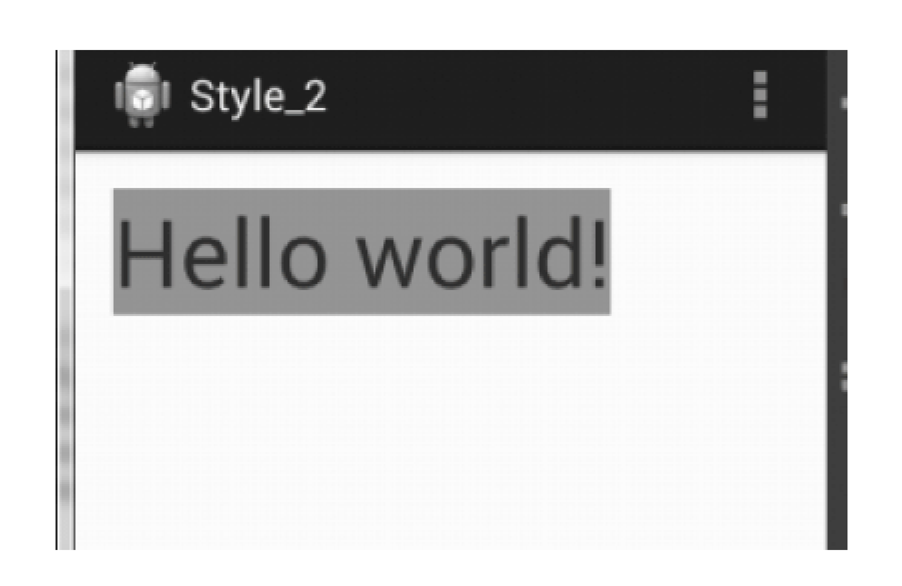


图 6-7　样式的继承效果

6.3.2　selector 状态列表

与 UI 界面美化有密切关系的是 statList 状态列表，selector 对控件状态的改变是通过 UI 图形来表达的，比如按钮有按下的状态、默认的状态及屏蔽状态等 UI 图形界面向用户展示控件的状态。

1. selector 概述

文字颜色状态列表 XML 配置文件存放在 res\color 文件夹下，此信息来自于 Android 官方的 guide 手册中的内容。

在 DOC 文档中依次展开 Application Resources→Resource Types→Color State List，可以在其中找到具体的使用方法。另外，创建文字颜色 selector 时只能手动进行配置，使用 ADT 的向导创建 XML 配置文件已经无效，如图 6-8 所示。

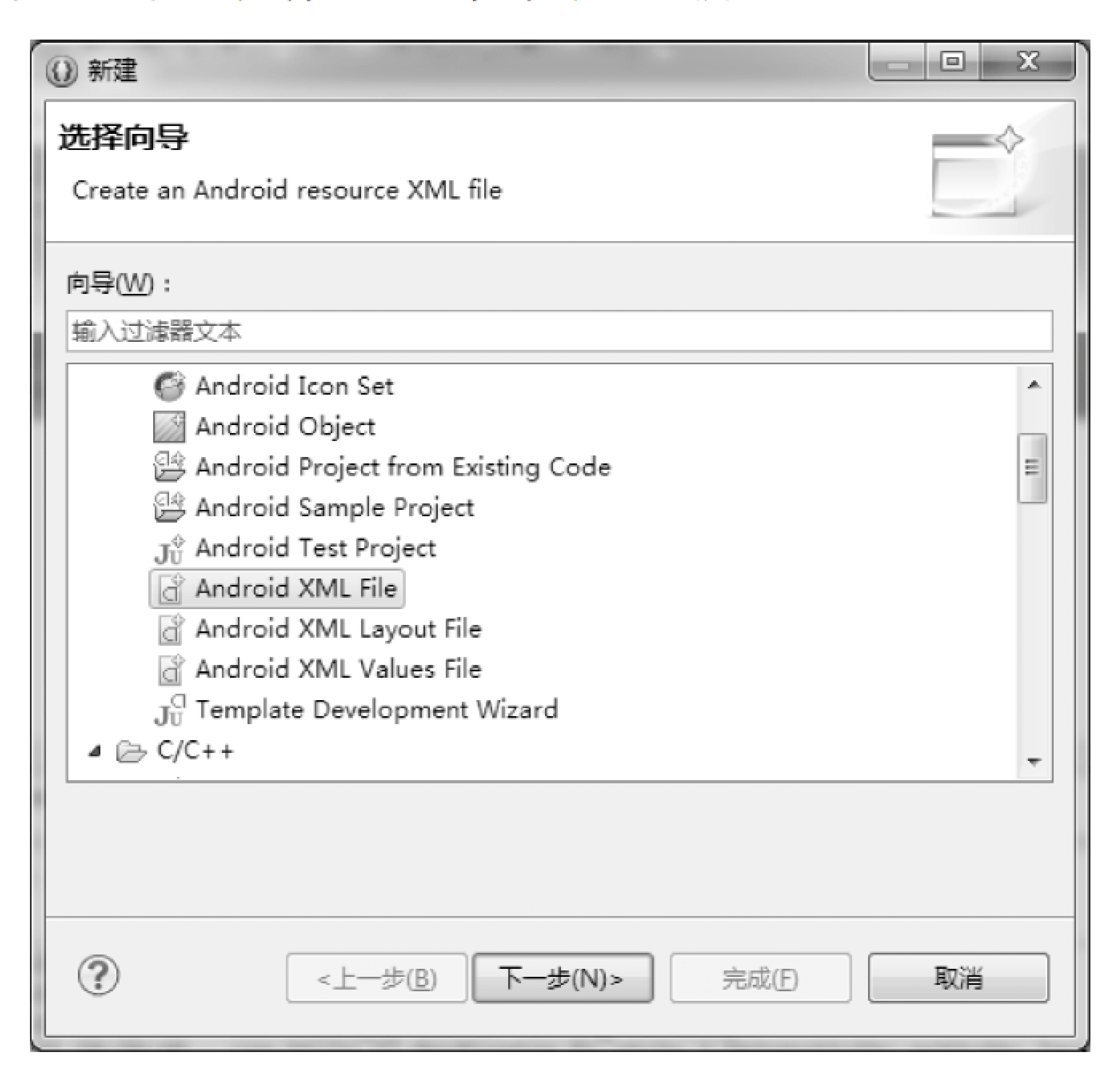


图 6-8　使用 XML 配置文件向导创建不了 selector 对象

值得注意的是，selector 状态列表 XML 配置文件的文件名 filename 即为 resourceId，引用文字颜色状态列表有两种方式，分别是在 Java 文件和 XML 文件中引用，引用方式如下。

- Java 引用方式：R. color. filename。
- XML 文件引用方式：@[package:]color/filename。

文字颜色状态列表 selector 的完整语法为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< selector xmlns:android = "http://schemas.android.com/apk/res/android">
 < item
      android:color = "hex_color"
      android:state_pressed = ["true"|"false"]
      android:state_focused = ["true"|"false"]
      android:state_selected = ["true"|"false"]
      android:state_checkable = ["true"|"false"]
      android:state_checked = ["true"|"false"]
      android:state_enabled = ["true"|"false"]
      android:state_window_focused = ["true"|"false"]>
</selector >
```

- <selector >元素：状态列表的根(root)结点名称为<selector>，必须要有的标签，用它来包含<item>元素。
- <item>元素：<item>元素是定义每种状态的详细信息，它是<selector>的子标签，而 android:color 属性是定义每种状态的文本颜色，值为十六进制的颜色代码，可以加入透明的 alpha 值，比如 Alpha-Red-Green-Blue，取值的格式为 #RGB、#ARGB、#RRGGBB 和 #AARRGGBB。

属性 android:state_pressed 的值为 true 和 false，代表控件按下和不按下的状态匹配。

属性 android:state_focused 的值为 true 代表获得了焦点，为 false 代表没有获得焦点。

属性 android:state_selected 的值为 true 代表控件被选中，为 false 代表没有被选中。

android:state_checkable 的值为 true 代表控件能被 checked，为 false 代表控件不能被 checked 的状态。

属性 android:state_checked 的值为 true 代表控件已被 checked，为 false 代表控件并没有被 checked 的状态。

属性 android:state_enabled 的值为 true 代表控件可以被使用，为 false 代表控件是不可用状态。

属性 android:state_window_focused 的值为 true 代表当前的窗体获得了焦点，为 false 代表窗体并没有获得焦点。

那么，文字颜色 selector 怎样应用呢？文字颜色 selector 是用来匹配每种状态 UI 变化的。

2. 文字颜色 selector

创建名称为 selectorView 的 Android 项目，在 res\color 文件夹下创建名称为 button_sele_text.xml 的文字颜色配置文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< selector xmlns:android = "http://schemas.android.com/apk/res/android">
```

```
    <item android:state_pressed = "true"
        android:color = "#FF0000" />
    <item android:state_focused = "true"
        android:color = "#00F00F" />
    <item android:color = "#0000FF" />
</selector>
```

以上文字颜色 selector 的功能就是定义控件在默认状态时的文字颜色为＃0000FF(蓝色)，而当 android:state_pressed＝"true"控件被按下时，颜色值为 android:color＝"＃FF0000"(红色)，获得焦点时，android:state_focused＝"true"，控件颜色为 android:color＝"＃00FF00"(绿色)。

在 main.xml 布局文件中的 Button 控件引用这个文字颜色 selector 资源，代码为：

```
<Button
    android:id = "@ + id/button1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_below = "@ + id/textView1"
    android:layout_centerHorizontal = "true"
    android:layout_marginTop = "86dp"
    android:text = "Button"
    android:textColor = "@color/sele_text"/>
```

6.3.3 背景图片 selector

在 6.3.2 节中介绍了文字颜色 selector 美化的实现，本节介绍一个与文字美化同样重要的功能——背景状态美化，即用漂亮的背景图片来表达控件的状态信息。

1. 背景图片 selector 概述

背景图片 selector 的作用是控件在不同的状态下显示出不同的背景图片，比如 Button 按钮被按下、抬起，默认时的背景图片都不一样，所以需要背景图片 selector 的美化。

想要实现背景图片 selector 的美化，XML 配置文件要存放在 res\drawable 文件夹下。可以在 Android 官方的 guide 手册的 Application Resources\Resource Types\Drawable 中的 State List 中找到。

如果想引用 selector 资源，只需要以 XML 文件名 filename 作为 resourceId 即可。引用的方式分别为 Java 方式和 XML 方式，例如：

- Java 引用方式：R.drawable.filename。
- XML 引用方式：@[package:]drawable/filename。

背景美化 selector 文件 ButView.xml 的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<selector xmlns:android = "http://schemas.android.com/apk/res/android"
    android:constantSize = ["true"|"false"]
    android:dither = ["true"|"false"]
    android:variablePadding = ["true"|"false"]>
    <item
      android:drawable = "@[package:]drawable/drawabe_resource"
      android:state_pressed = ["true"|"false"]
      android:state_focused = ["true"|"false"]
```

```
        android:state_selected = ["true"|"false"]
        android:state_checkable = ["true"|"false"]
        android:state_checked = ["true"|"false"]
        android:state_enabled = ["true"|"false"]
        android:state_window_focused = ["true"|"false"]/>
</selector>
```

和文字颜色 elector 的语法相似，仅仅在这里定义 android：drawable 属性，而不是 android：color。

在控件中使用的实例代码为：

```
android:background = "@drawable/buttonSelect"
```

2. 使用 selector 美化按钮控件

在大多数 UI 控件的美化过程中，都是将文字颜色与背景图片 selector 进行联合使用，本实例来实现 Button 控件的美化。

在 Android 中新建一个名为 UICotrol 的 Android 项目实现美化 UI 的功能，具体实现步骤如下：

（1）在文件夹 res\drawable-hdpi 下添加 3 个图片资源，名称分别为 btn_style_focused.9、btn_style_normal.9 和 btn_style_pressed.9。

（2）在 res 文件夹下创建一个 color 文件，再创建一个名为 button_sele_text.xml 的文件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< selector xmlns:android = "http://schemas.android.com/apk/res/android">
    < item android:state_pressed = "true"
        android:color = "#FF0000" />
    < item android:state_focused = "true"
        android:color = "#00F00F" />
    < item android:color = "#0000FF" />
</selector>
```

（3）在 res 下创建一个名为 drawable 的文件夹，在该文件夹中创建一个名为 button_sele_drawable.xml 的文件，用于为 Button 创建背景图片。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< selector xmlns:android = "http://schemas.android.com/apk/res/android">
    < item android:state_pressed = "true"
          android:drawable = "@drawable/btn_style_pressed" />
    < item android:state_focused = "true"
        android:drawable = "@drawable/btn_style_focused" />
    < item android:drawable = "@drawable/btn_style_normal" />
</selector>
```

（4）打开 res\layout 目录下的 main.xml 文件，在 id 为 button2 的 Button 控件中应用文字颜色和背景图片 selector 资源。代码为：

```
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
```

```
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/textView1"
        android:layout_centerHorizontal = "true"
        android:layout_marginTop = "86dp"
        android:text = "Button"
        android:textColor = "@color/button_sele_text"/>
</RelativeLayout >
```

运行程序,效果如图 6-9 所示。

当单击按钮时,Button 由默认的蓝色变为红色,效果如图 6-10 所示。

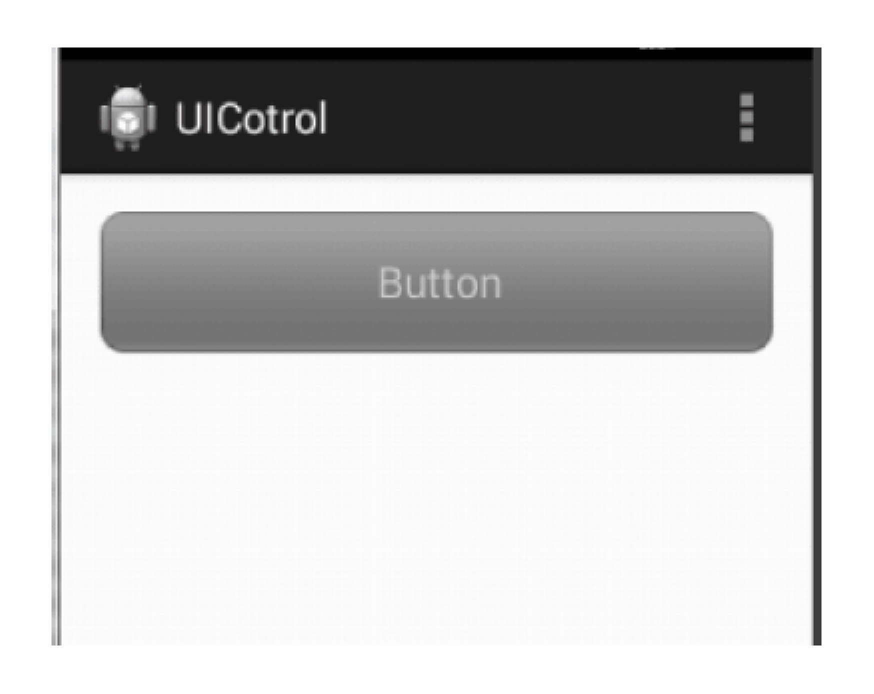


图 6-9　Button 的默认效果

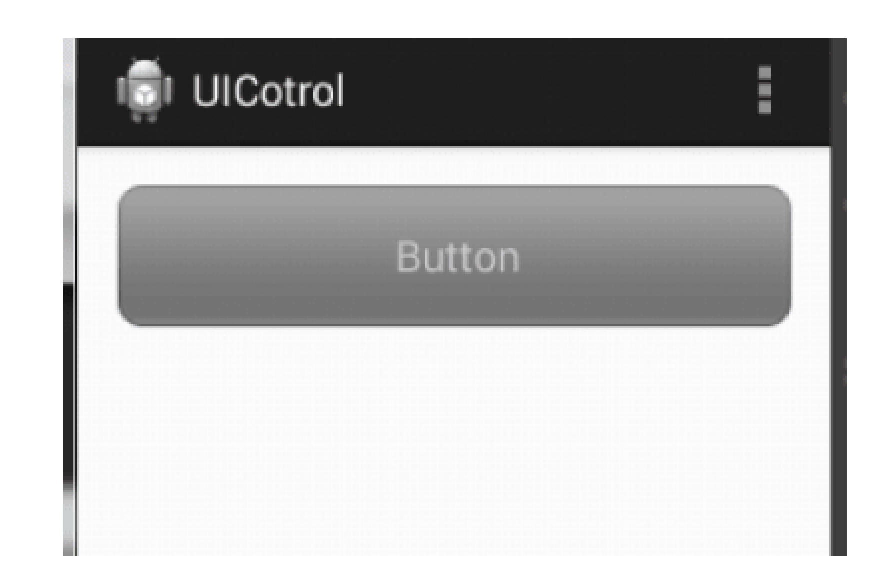


图 6-10　按下 Button 的效果

6.4　Android 动画

在 Android 中主要有两种动画表现方式,即补间动画 Tween Animation 和逐帧动画 Frame Animation。逐帧动画主要用于游戏开发,在应用程序的开发中用得比较少。

6.4.1　补间动画

补间动画 Tween Animation 是 Android 中表现动画的主要方式,例如对控件添加动画、在 Activity 之间切换时添加动画等,Tween Animation 动画的 XML 使用格式为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
    android:interpolator = "@[package:]anim.interpolator_resource"
    android:shareInterpolator = ["true"|"false"]>
    < alpha android:fromAlpha = "float"android:toAlpha = "float"/>
    < scale android:fromXScale = "float" android:toXScale = "float"
        android:fromYScale = "float"android:toXScale = "float"
        android:pivotY = "float"/>
```

```
    <translate android:fromX = "float"android:toX = "float"
        android:fromY = "float"android:toY = "float"/>
    <rotate android:fromDegrees = "float"android:toDegrees = "float"
        android:pivotX = "float"android:pivotY = "float"/>
    <set>…
        </set>
</set>
```

从 XML 配置文件中可以看到，补间动画支持 Alpha 透明、Scale 缩放、Translate 移动和 Rotate 旋转等。

1. Alpha 透明

在 Android 中提供了 Alpha 对象用于实现图像的淡出淡入效果，主要控制的是透明度的变化。在 Alpha 中定义的可实现淡入淡出效果的关键属性如下。

- fromAlpha：表示动画起始时的透明度，0.0 表示完全透明，1.0 表示完全不透明，其取值范围为 0.0～1.0 的 float 数据类型的数字。
- toAlpha：表示动画结束时的透明度，其取值范围为 0.0～1.0 的 float 数据类型的数字。
- duration：动画持续的时间，单位为毫秒。

下面的实例用于实现单击改变图片的透明度。

【例 6-5】 改变图片透明度实例。

本例首先显示一幅透明度为 100 的图片，单击该图片区域内的任意处，将改变该图片的透明度，并在界面中显示更改透明度后的图片。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li6_5Alpha。

(2) 打开 res\layout 目录下的布局文件 main.xml，该文件首先设置 LinearLayout 的摆放顺序，然后需要对 ImageView 的属性进行设置。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1" >
    <ImageView
    android:id = "@ + id/ImageView01"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:paddingLeft = "30dip"
    android:paddingRight = "30dip"
    android:visibility = "visible"/>
</LinearLayout>
```

(3) 打开 src\fs.li6_5alpha 包下的 MainActivity.java 文件，用于改变图片的透明度。代码为：

```
package fs.li6_5alpha;
import android.app.Activity;
import android.os.Bundle;
```

```
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageView;
public class MainActivity extends Activity
{
     ImageView iv;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);            //调用父类
        setContentView(R.layout.main);
        //初始化 ImageView
        iv = (ImageView)findViewById(R.id.ImageView01);
        //为 ImageView 设置图片
        iv.setImageDrawable(getResources().getDrawable(R.drawable.big));
        iv.setAlpha(100);                              //设置透明度为 100
        //为 ImageView 设置监听,当单击图片的时候,图片的透明度增加
        iv.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v)
                    {
iv.setImageDrawable(getResources().getDrawable(R.drawable.big));
                        iv.setAlpha(255);              //设置透明度为 255
                    }
                }
        );
    }
}
```

运行程序,效果如图 6-11 所示,单击屏幕,效果如图 6-12 所示。



图 6-11　透明度为 100 时



图 6-12　透明度为 255 时

2. 图像缩放 Scale

在 Android 中提供了 Scale 对象用于实现图像的缩放。在 Scale 中定义的可实现缩放的关键属性如下。

- fromXScale：起始时 X 坐标的尺寸，设置为 1.0 说明是整个图片 X 轴的长度。
- toXScale：结束时 X 坐标的尺寸，设置为 0.0 说明整个图片 X 轴完全收缩到无。
- fromYScale：起始时 Y 坐标的尺寸，设置为 1.0 说明是整个图片 Y 轴的长度。
- toYScale：结束时 Y 坐标的尺寸，设置为 1.0 说明在收缩时 Y 轴的长度保持不变。
- pivotX：动画在 X 轴方向缩放的中心点，取值为 0%～100%或 0%p～100%p，50%为相对于自己的中心位置，50%p 表示相对于父控件的中心位置，也就是 p(parent)的意义。
- pivotY：动画在 Y 轴方向缩放的中心点，取值为 0%～100%或 0%p～100%p，50%为相对于自己的中心位置，50%p 表示相对于父控件的中心位置，也就是 p(parent)的意义。
- duration：设置动画执行时间。
- interpolator：指定一个动画的插入器。interpolator 定义一个动画的变化率，这使得基本的动画效果（Alpha、Scale、Translate、Rotate）得以加速、减速、重复等。Android 提供了几个 Interpolator 子类，实现了不同的速度曲线。
  ◇ linear_interpolator：使动画以均匀的速率改变。
  ◇ cycle_interolator：动画循环播放特定的次数，速率改变沿着正弦曲线。
  ◇ accelerate_decelerate_interpolator：在动画开始与结束的地方速率改变比较慢，在中间时加速。
  ◇ accelerate_interpolator：在动画开始的地方速率改变比较慢，然后开始加速。
  ◇ decelerate_interpolator：在动画开始的地方速率改变比较慢，然后开始减速。
- zAdjustment：定义动画的 Z 轴方向的位置，值为 normal 保持位置不变，为 top 保持在最上层，为 bottom 保持在最下层。
- repeatCount：动画的重复次数。
- repeatMode：定义重复的模式，其中值为 restart 表示重新开始，为 reverse 表示先倒退再执行，倒退也算一次。
- startOffset：动画之间的时间间隔，即从上次动画停多少时间开始执行下一个动画。

下面通过一个实例来演示图像的缩放效果。

【例 6-6】 演示图像的缩放效果。

本例主要用于实现：当单击界面中的“放大”按钮时，图像会增大，当单击界面中的“缩小”按钮时，图像会变小。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li6_6Scale。

(2) 打开 res\layout 目录下的 main.xml 文件，总的布局方式为帧布局，摆放方向为横向，然后设置 LinearLayout 布局，并在该布局中设置两个 Button 按钮的位置，再设置另一个 LinearLayout 布局，并在该布局中设置 ImageView 的具体属性。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/bj1"
```

```
    android:orientation = "vertical" >
    <LinearLayout
        android:id = "@ + id/LinearLayout01"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_gravity = "center"
        android:gravity = "left|bottom"
        android:orientation = "horizontal" >
        <Button
            android:id = "@ + id/Button2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center"
            android:layout_weight = "4.04"
            android:text = "放大"
            android:textSize = "25dip" />
        <Button
            android:id = "@ + id/Button1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "缩小"
            android:textSize = "25dip" />
    </LinearLayout>
    <LinearLayout
    android:id = "@ + id/LinearLayout03"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content">
    <ImageView
        android:id = "@ + id/ImageView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    </LinearLayout>
</FrameLayout>
```

(3) 打开 src\fs.li6_6scale 目录下的 MainActivity.java 文件,实现当单击屏幕中的“放大”和“缩小”按钮时放大和缩小图片的功能。代码为:

```
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
public class MainActivity extends Activity
{
    ImageView iv;                                   //声明 ImageView 的引用
    Bitmap bmp;                                     //声明 Bitmap 的引用
    int screenWidth;                                //屏幕宽度
    int secrenHeight;                               //屏幕高度
    Button b2;
```

```java
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);                    //切换到主界面
    //创建 Bitmap 对象
    bmp = BitmapFactory.decodeResource(getResources(), R.drawable.dcuk2);
    DisplayMetrics dm = new DisplayMetrics();         //创建矩阵
    getWindowManager().getDefaultDisplay().getMetrics(dm);
    screenWidth = dm.widthPixels;                     //得到屏幕的宽度
    screenWidth = dm.heightPixels - 80;               //得到屏幕的高度
    iv = (ImageView)findViewById(R.id.ImageView1);    //ImageView 控件
    Button b1 = (Button)findViewById(R.id.Button1);   //"缩小"按钮
    b2 = (Button)findViewById(R.id.Button2);          //"放大"按钮
    iv.setImageBitmap(bmp);                           //为 ImageView 设置图片
    b1.setOnClickListener                             //设置监听
    (
        new OnClickListener()
        {
            public void onClick(View v)
            {
                //单击"缩小"按钮,图像缩小到原来的 0.6 倍
                iv.setImageBitmap(scaleToFit(bmp,0.8f));
                bmp = scaleToFit(bmp,0.8f);                   }
        }
    );
    b2.setOnClickListener//设置监听
    (
        new OnClickListener()
        {
            public void onClick(View v)
            {
                //单击"放大"按钮,图像放大到原来的 1.5 倍
                iv.setImageBitmap(scaleToFit(bmp,1.5f));
                bmp = scaleToFit(bmp,1.5f);
            }
        }
    );
}
public static Bitmap scaleToFit(Bitmap bm,float scale)   //缩放图片的方法
{
    Bitmap bmResult = null;
    if((bm.getWidth()< 280)&&(bm.getWidth()> 0)&&(bm.getHeight()> 0))
    {
        int width = bm.getWidth();                    //图片宽度
        int height = bm.getHeight();                  //图片高度
        Matrix matrix = new Matrix();                 //创建矩阵
        matrix.postScale(scale, scale);               //图片等比例缩小为原来的 fblRatio 倍
        //声明位图
        bmResult = Bitmap.createBitmap(bm, 0, 0, width, height, matrix, true);
    }
    else
    {
        System.out.println();
```

```
            }
            return bmResult;
        }
    }
```

运行程序,效果如图 6-13 所示。

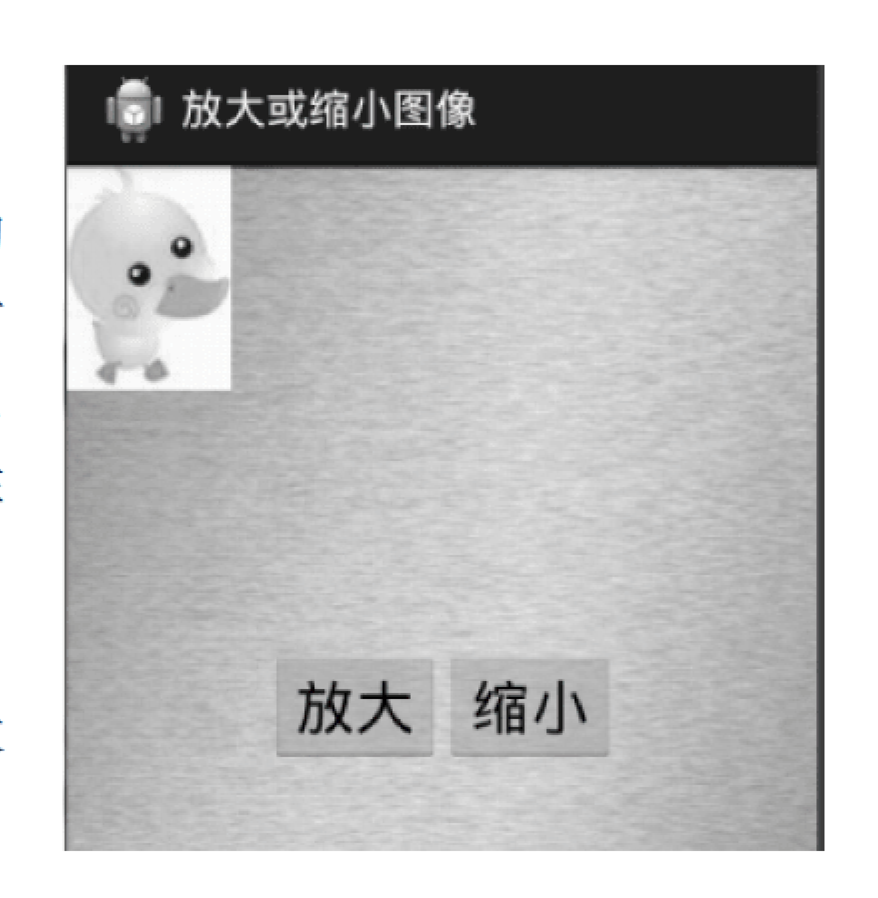


图 6-13 图像的缩放效果

3. 图像旋转

在 Android 中提供了 Rotate 对象用于实现图像的旋转。旋转和缩放都需要指定中心点,同样在取值时要指定相对于父控件还是相对于自己,50%表示自己的中心。在 Rotate 中定义的可实现旋转的关键属性如下。

- fromDegrees:动画开始时的角度。
- toDegrees:动画结束时旋转的角度,可以大于 360°。

下面通过一个实例来说明图像的旋转。

【例 6-7】 图像旋转实例。

本例实现在主界面中单击"向右旋转"按钮即可将图片向右旋转,单击"向左旋转"按钮即可将图像向左旋转。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 li6_7Rotate。

(2) 打开 res\layout 目录下的 main.xml 文件,总的布局为帧布局,摆放方向为横向,然后设置 LinearLayout 布局,并在该布局中设置两个 Button 按钮的位置,再设置另一个 LinearLayout 布局,并在该布局中设置 ImageView 的具体属性。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1">
    < LinearLayout
     android:id = "@ + id/LinearLayout01"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent"
     android:orientation = "horizontal"
     android:gravity = "left|bottom">
        < Button
            android:id = "@ + id/Button1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "向左旋转"
            android:textSize = "25dip" />
        < Button
        android:text = "向右旋转"
        android:textSize = "25dip"
        android:id = "@ + id/Button2"
```

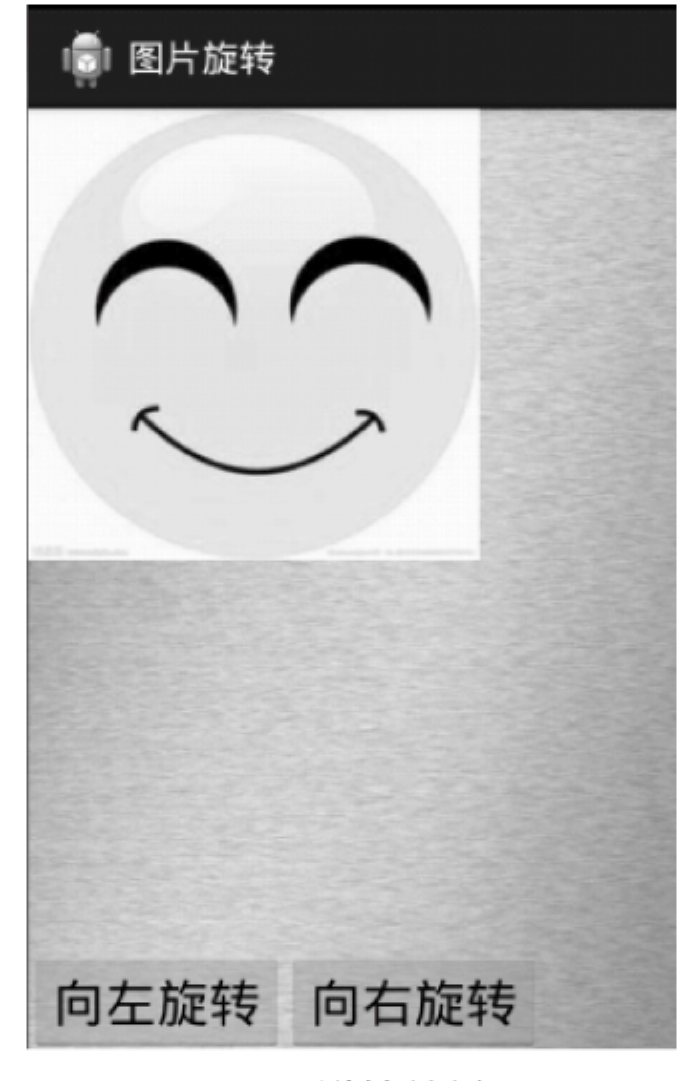
```
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    </LinearLayout>
    <LinearLayout
    android:id = "@ + id/LinearLayout3"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content">
        < ImageView
        android:id = "@ + id/ImageView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    </LinearLayout>
</FrameLayout>
```

(3) 打开 src\fs.li6_7rotate 包下的 MainActivity.java 文件，在文件中实现图像的向左旋转和向右旋转。代码为：

```
package fs.li6_7rotate;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
public class MainActivity extends Activity
{
    ImageView iv;                                      //声明 ImageView
    Bitmap bmp;                                        //声明 Bitmap
    int screenWidth;                                   //屏幕宽度
    int secrenHeight;                                  //屏幕高度
    float scaleWidth = 1f;                             //屏幕比例
    float scaleHeight = 1f;                            //屏幕高度
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //创建 Bitmap
        bmp = BitmapFactory.decodeResource(getResources(), R.drawable.face);
        DisplayMetrics dm = new DisplayMetrics();          //创建显示矩阵
        getWindowManager().getDefaultDisplay().getMetrics(dm);
        screenWidth = dm.widthPixels;
        screenWidth = dm.heightPixels - 60;
        iv = (ImageView)findViewById(R.id.ImageView1);  //获取 ImageView
        Button b1 = (Button)findViewById(R.id.Button1); //左转按钮
        Button b2 = (Button)findViewById(R.id.Button2); //右转按钮
        iv.setImageBitmap(bmp);                         //为 ImageView 设置图片
        b1.setOnClickListener                           //设置监听
        (
                new OnClickListener()
```

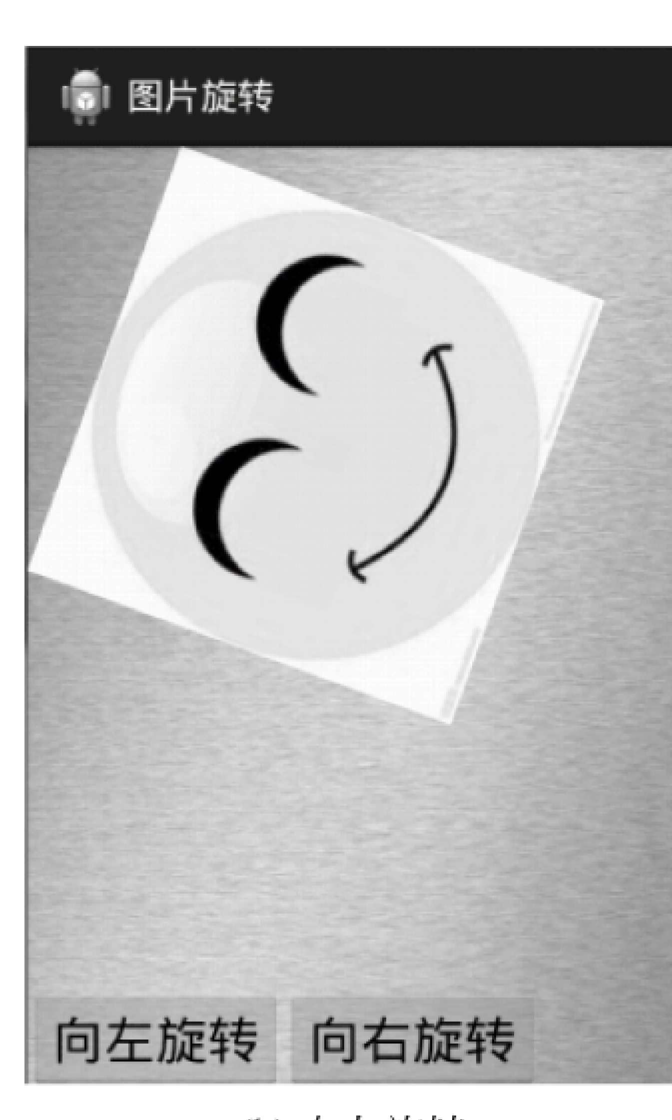
```
                {
                    public void onClick(View v) {
                        iv.setImageBitmap(rotateToFit(bmp, - 70f));   //左转
                        bmp = rotateToFit(bmp, - 70f);
                    }
                }
            );
            b2.setOnClickListener                                       //设置监听
            (
                new OnClickListener()
                {
                    public void onClick(View v) {
                        iv.setImageBitmap(rotateToFit(bmp,70f));   //右转
                        bmp = rotateToFit(bmp,70f);
                    }
                }
            );
        }
        public static Bitmap rotateToFit(Bitmap bm,float degrees)       //缩放图片的方法
        {
            int width = bm.getWidth();                                  //图片宽度
            int height = bm.getHeight();                                //图片高度
            Matrix matrix = new Matrix();
            matrix.postRotate(degrees);
            //声明位图
            Bitmap bmResult = Bitmap.createBitmap(bm, 0, 0, width, height, matrix, true);
            return bmResult;
        }
    }
```

运行程序,效果如图 6-14(a)所示。当单击“向左旋转”按钮时,效果如图 6-14(b)所示;当单击“向右旋转”按钮时,效果如图 6-14(c)所示。





(a) 默认效果



(b) 向左旋转



(c) 向右旋转

图 6-14　图像的旋转效果

4. 图像平移

在 Android 中提供了 Translate 对象用于实现图像的平移。在 Translate 中定义的可实现平移的关键属性如下。

- fromXDelta：动画在 X 轴方向的起始坐标。
- toXDelta：动画在 X 轴方向的结束坐标。
- formYDelta：动画在 Y 轴方向的起始坐标。
- toYDelta：动画在 Y 轴方向的结束坐标。

下面通过一个实例来实现补间动画。

【例 6-8】 综合实现图像的 4 种动画效果。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li6_8Tween。

(2) 打开 res\layout 目录下的 main.xml 文件，代码为：

```
<! -- 动画 MM -->
< ImageView
apk:id = "@ + id/TweenMM"
apk:src = "@drawable/hai"
apk:layout_width = "wrap_content"
apk:layout_height = "wrap_content"/>
<! -- 动画控制按钮 -->
< LinearLayout
apk:layout_weight = "1"
apk:orientation = "horizontal"
apk:layout_width = "fill_parent"
apk:layout_height = "wrap_content">
< Button
apk:text = "改变大小"
apk:layout_weight = "1"
apk:layout_width = "fill_parent"
apk:layout_height = "wrap_content"
apk:onClick = "onBtnScaleAnimClick"/>
< Button
apk:text = "淡入淡出"
apk:layout_weight = "1"
apk:layout_width = "fill_parent"
apk:layout_height = "wrap_content"
apk:onClick = "onBtnAlphaAnimClick"/>
</LinearLayout >
< LinearLayout
apk:layout_weight = "1"
apk:orientation = "horizontal"
apk:layout_width = "fill_parent"
apk:layout_height = "wrap_content">
< Button
apk:text = "位置移动"
apk:layout_weight = "1"
apk:layout_width = "wrap_content"
apk:layout_height = "wrap_content"
apk:onClick = "onBtnTranslateAnimClick"/>
< Button apk:text = "旋转"
```

```
apk:layout_weight = "1"
apk:layout_width = "wrap_content"
apk:layout_height = "wrap_content"
apk:onClick = "onBtnRotateAnimClick"/>
</LinearLayout>
</LinearLayout>
```

(3) 在 res\layout 目录下分别建立名为 main_rotate. xml、main_scale. xml、main_translate. xml、main_alpha. xml 文件。

main_alpha. xml 的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    < alpha
        android:fromAlpha = "0.1"
        android:toAlpha = "1.0"
        android:duration = "2000"   />
</set>
```

main_scale. xml 的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    < scale
        android:interpolator = "@android:anim/accelerate_decelerate_interpolator"
        android:fromXScale = "0.0"
        android:toXScale = "1.0"
        android:fromYScale = "0.0"
        android:toYScale = "1.0"
        android:pivotX = "80%"
        android:pivotY = "50"
        android:fillAfter = "true"
        android:duration = "1000" />
</set>
```

main_translate. xml 的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    < translate
        android:fromXDelta  =  "10"
        android:toXDelta  =  "100"
        android:fromYDelta  =  "10"
        android:toYDelta  =  "100"
        android:duration  =  "1000" />
</set>
```

main_rotate. xml 的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    < rotate
        android:interpolator = "@android:anim/accelerate_decelerate_interpolator"
        android:fromDegrees = "0"
```

```
        android:toDegrees = "360"
        android:pivotX = "0"
        android:pivotY = "50 %"
        android:duration = "1000" />
</set>
```

(4) 打开 src\fs.li6_8tween 包下的 MainActivity.java 文件,代码为:

```
package fs.li6_8tween;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import fs.li6_8tween;
//通过 XML 配置文件的方式实现 Tween 动画
public class MainActivity extends Activity
{
    public static final String TAG = "TweenActivity";
    //动画图片
    private ImageView tweenMM;
    /*
      ** @see android.app.Activity#onCreate(android.os.Bundle)
      */
    public void onCreate(Bundle cycle)
    {
        super.onCreate(cycle);
        super.setContentView(R.layout.main);
        //取得动画图片
        this.tweenMM = (ImageView) super.findViewById(R.id.TweenMM);
    }
    //按钮: 尺寸变化动画
    public void onBtnScaleAnimClick(View view)
    {
        //动画开始
        this.doStartAnimation(R.layout.main_scale);
    }
    //按钮: 渐变动画
    public void onBtnAlphaAnimClick(View view)
    {
        //动画开始
        this.doStartAnimation(R.layout.main_alpha);
    }
    //按钮: 位置变化动画
    public void onBtnTranslateAnimClick(View view)
    {
        //动画开始
        this.doStartAnimation(R.layout.main_translate);
    }
    //按钮: 旋转动画
    public void onBtnRotateAnimClick(View view)
    {
        //动画开始
```

```
            this.doStartAnimation(R.layout.main_rotate);
        }
         //开始动画
        private void doStartAnimation(int animId)
        {
            //加载动画
            Animation animation = AnimationUtils.loadAnimation(this, animId);
            //动画开始
            this.tweenMM.startAnimation(animation);
        }
    }
```

运行程序,效果如图 6-15 所示。



图 6-15 Tween 动画效果

5. 图像扭曲

在 Android 中提供了 drawBitmapMesh 类实现图像的扭曲,Canvas 的 drawBitmapMesh 定义如下:

```
public void drawBitmapMesh(Bitmap bitmap, int meshWidth,
int meshHeight, float [ ] verts, int vertOffset, int [ ]
colors, int colorOffset, Paint paint)
```

其用于表示将图像绘制在网格上,可以将画板想象成一张格子布,在这张布上绘制图像。对于一个网格端点均匀分布的网格来说,横向有 meshWidth+1 个顶点,纵向有 meshHeight+1 个端点。顶点数组 verts 是以行优先的数组(二维数组以一维数组表示,先行后列)。网格可以不均匀分布,参数定义如下。

- Bitmap:需要绘制在网格上的图像。
- meshWidth:网格的宽度方向的数目(列数),为 0 时不绘制图像。
- meshHeight:网格的高度方向的数目(含数),为 0 时不绘制图像。
- verts:为(x,y)对的数组,表示网格顶点的坐标,至少需要有"(meshWidth+1) * (meshHeight+1) * 2 + meshOffset"个(x,y)坐标。
- vertOffset:用于控制 verts 数组中开始跳过的(x,y)对的数目。
- Colors:可以为空,不为空为没个顶点定义对应的颜色值,至少需要有"(meshWidth+1) * (meshHeight+1) * 2 + meshOffset"个(x,y)坐标。
- colorOffset:colors 数组中开始跳过的(x,y)对的数目。
- paint:可以为空。

值得注意的是,当程序希望调用 drawBitmapMesh 方法对位图进行扭曲时,关键是计算 verts 数组的值——该数组的值记录了扭曲后的位图上各"顶点"的坐标。

下面的实例将会通过 drawBitmapMesh 方法来控制图片的扭曲。

【例 6-9】 通过 drawBitmap Mesh 方法控制图片的扭曲。

本例实现:当用户"触摸"图片的指定点时,该图片会在这个点被用户"按"下去,就像将这张图片铺在"极软的床上"一样。

为了实现这个效果,代码要在用户触摸图片的指定点时动态地改变 verts 数组中每个元素的位置(控制扭曲后每个顶点的坐标),这种改变也简单:程序计算图片上每个顶点与

触摸点的距离，顶点与触摸点的距离越小，该顶点向触摸点移动的距离越大。

其实现步骤如下：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 li6_9drawBitmapMesh。

(2) 打开 res\layout 目录下的 main.xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/hello_world"/>
</LinearLayout >
```

(3) 编写 Activity 文件。打开 src/fs. li6_9drawbitmapmesh 包下的 MainActivity.java 文件，将代码修改为：

```
public class MainActivity extends Activity
{
    private Bitmap bitmap;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this , R.drawable.ch));
    }
    private class MyView extends View
    {
        //定义两个常量,这两个常量指定该图片横向、纵向上都被划分为 50 格
        private final int WIDTH = 50;
        private final int HEIGHT = 50;
        //记录该图片上包含 441 个顶点
        private final int COUNT = (WIDTH + 1) * (HEIGHT + 1);
        //定义一个数组,保存 Bitmap 上的 21 * 21 个点的坐标
        private final float[] verts = new float[COUNT * 2];
        //定义一个数组,记录 Bitmap 上的 21 * 21 个点经过扭曲后的坐标
        //对图片进行扭曲的关键就是修改该数组中元素的值
        private final float[] orig = new float[COUNT * 2];
        public MyView(Context context , int drawableId)
        {
            super(context);
            setFocusable(true);
            //根据指定资源加载图片
            bitmap = BitmapFactory.decodeResource(getResources(),
                drawableId);
            //获取图片的宽度、高度
            float bitmapWidth = bitmap.getWidth();
            float bitmapHeight = bitmap.getHeight();
            int index = 0;
            for (int y = 0; y <= HEIGHT; y++)
            {
```

```
            float fy = bitmapHeight * y / HEIGHT;
            for (int x = 0; x <= WIDTH; x++)
            {
                float fx = bitmapWidth * x / WIDTH;
                //初始化 orig、verts 数组
        //初始化后,orig、verts 两个数组均匀地保存了 21 * 21 个点的 X、Y 坐标
                orig[index * 2 + 0] = verts[index * 2 + 0] = fx;
                orig[index * 2 + 1] = verts[index * 2 + 1] = fy;
                index += 1;
            }
        }
        //设置背景色
        setBackgroundColor(Color.WHITE);
    }
    @Override
    protected void onDraw(Canvas canvas)
    {
    //从第 1 个点(由第 5 个参数 0 控制)开始对 bitmap 按 verts 数组进行扭曲
    canvas.drawBitmapMesh(bitmap, WIDTH, HEIGHT, verts, 0, null, 0, null);
    }
    //工具方法,用于根据触摸事件的位置计算 verts 数组中各元素的值
    private void warp(float cx, float cy)
    {
        for (int i = 0; i < COUNT * 2; i += 2)
        {
            float dx = cx - orig[i + 0];
            float dy = cy - orig[i + 1];
            float dd = dx * dx + dy * dy;
            //计算每个坐标点与当前点(cx,cy)之间的距离
            float d = (float)Math.sqrt(dd);
            //计算扭曲度,距离当前点(cx,cy)越远,扭曲度越小
            float pull = 80000 / ((float) (dd * d));
            //对 verts 数组(保存 bitmap 上 21 * 21 个点经过扭曲后的坐标)重新赋值
            if (pull >= 1)
            {
                verts[i + 0] = cx;
                verts[i + 1] = cy;
            }
            else
            {
                //控制各顶点向触摸事件发生点偏移
                verts[i + 0] = orig[i + 0] + dx * pull;
                verts[i + 1] = orig[i + 1] + dy * pull;
            }
        }
        //通知 View 组件重绘
        invalidate();
    }
    @Override
    public boolean onTouchEvent(MotionEvent event)
    {
```

```
                //调用 warp 方法根据触摸屏事件的坐标点来扭曲 verts 数组
                warp(event.getX(), event.getY());
                return true;
            }
        }
    }
```

运行程序,单击界面中显示的图片,效果如图 6-16 所示。

图 6-16 图像扭曲效果

6.4.2 逐帧动画

Frame 动画主要是通过 AnimationDrawable 类实现的,它有 start()和 stop()两个重要的方法来启动和停止动画。Frame 动画一般通过 XML 文件配置,在工程的 res\anim 目录下创建一个 XML 配置文件,该配置文件有一个<animation-list>根元素和若干个<item>子元素。定义逐帧动画的语法格式为:

```
[html] view plaincopyprint?<?xml version = "1.0" encoding = "utf - 8"?>
< animation - list xmlns:android = "http://schemas.android.com/apk/res/android"
    android:oneshot = ["true" | "false"] >
    < item
        android:drawable = "@[package:]drawable/drawable_resource_name"
        android:duration = "integer" />
</animation - list >
```

值得注意的是:

- <animation-list>元素是必需的,并且必须作为根元素,可以包含一或多个<item>元素; android:onshot 如果定义为 true,此动画只会执行一次,如果为 false,则一直循环。
- <item>元素代表一帧动画; android:drawable 指定此帧动画所对应的图片资源; android:druation 代表此帧持续的时间,整数,单位为毫秒。

下面通过一个实例来演示逐帧动画的使用。

【例 6-10】 一个卡通"滚动"的逐帧(Frame)动画。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 li6_9more Frame。

(2) 把 j1、j2、j3、j4、j5 这 5 张图片放到 res\drawable 目录下。

(3) 在 res 根目录下新建一个 anim 子目录文件。实现操作为:选择 res 并右击,在弹出的快捷菜单中选择"新建"下的"文件夹"命令,在"文件夹"右侧的文本框中输入"anim",然后单击"完成"按钮。

(4) 在 res\anim 目录下创建一个 face.xml 文件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< animation - list
xmlns:apk = "http://schemas.android.com/apk/res/android"
apk:oneshot = "false">
 < item apk:drawable = "@drawable/j1"
 apk:duration = "500" />
 < item apk:drawable = "@drawable/j2"
```

```
 apk:duration = "500" />
 < item apk:drawable = "@drawable/j3"
 apk:duration = "500" />
 < item apk:drawable = "@drawable/j4"
 apk:duration = "500" />
 < item apk:drawable = "@drawable/j5"
 apk:duration = "500" />
</animation - list >
```

（5）打开 res\layout 目录下的 main. xml 文件，将代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
xmlns:apk = "http://schemas.android.com/apk/res/android"
apk:orientation = "vertical"
apk:layout_width = "fill_parent"
apk:layout_height = "fill_parent">
<! -- Frame 动画图片 -- >
< ImageView
apk:id = "@ + id/ImgDance"
apk:layout_width = "wrap_content"
apk:layout_height = "wrap_content"
apk:background = "@anim/face" />
<! -- 动画控制按钮 -- >
< LinearLayout
apk:layout_width = "fill_parent"
apk:layout_height = "wrap_content"
apk:orientation = "horizontal">
 < Button
 apk:text = "开始"
 apk:layout_width = "wrap_content"
 apk:layout_height = "wrap_content"
 apk:onClick = "onStartDance" />
 < Button
 apk:text = "结束"
 apk:layout_width = "wrap_content"
 apk:layout_height = "wrap_content"
 apk:onClick = "onStopDance" />
 </LinearLayout > </LinearLayout >
```

（6）打开 src\com. example. frame_e 包下的 MainActivity. java 文件，将代码修改为：

```
package fs.li6_9moreframe;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import com.example. frame_e.R;
//Frame 动画
public class MainActivity extends Activity
{
    public static final String TAG = "MainActivity";
    //显示动画的控件
    private ImageView imgDance;
```

```
    //Frame 动画
    private AnimationDrawable animDance;
    public void onCreate(Bundle cycle)
    {
        super.onCreate(cycle);
        super.setContentView(R.layout.main);
        //实例化控件
        this.imgDance = (ImageView) super.findViewById(R.id.ImgDance);
        //获得背景(5 个图片形成的动画)
        this.animDance = (AnimationDrawable) this.imgDance.getBackground();
    }
    //按钮: 开始"滚动"动画
    public void onStartDance(View view)
    {
        this.animDance.start();
    }
    //按钮: 停止"滚动"动画
    public void onStopDance(View view)
    {
        this.animDance.stop();
    }
}
```

运行程序,效果如图 6-17 所示。

图 6-17　Frame 动画效果

6.5　图形与动画综合实例

下面通过一个综合实例实现野猪奔跑的动画。

【例 6-11】　在 Android 中实现野猪奔跑的动画。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 runpig。

(2) 在新建项目的 res 目录中创建一个名称为 anim 的文件夹,并在该文件夹中创建实现野猪做向右奔跑动作和向左奔跑动作的逐帧动画资源文件。创建文件夹的操作为:

选中 res 文件并右击,在弹出的快捷菜单中选择"新建→文件夹"命令,弹出如图 6-18 所示的"新建文件夹"对话框,在"文件夹名"文本框中输入对应的新建文件夹名称。

① 在 anim 文件夹中创建名称为 right.xml 的 XML 资源文件,在该文件中定义一个野猪做向右奔跑动作的动画,该动画由两帧组成,即由两个预先定义好的图片组成。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< animation - list xmlns:android = "http://schemas.android.com/apk/res/android" >
    < item android:drawable = "@drawable/pig1" android:duration = "30" />
    < item android:drawable = "@drawable/pig2" android:duration = "30" />
</animation - list >
```

② 在 anim 文件夹中创建名称为 left.xml 的 XML 资源文件,在该文件中定义一个野猪做向左奔跑动作的动画,该动画由两帧组成,即由两个预先定义好的图片组成。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
```



图 6-18 “新建文件夹”对话框

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/pig3" android:duration="30" />
    <item android:drawable="@drawable/pig4" android:duration="30" />
</animation-list>
```

(3) 在 amin 文件夹中，创建实现野猪向右奔跑和向左奔跑的补间动画资源文件。

① 在 anim 文件夹中创建名称为 rotright.xml 的 XML 资源文件，在文件中定义一个实现野猪向右奔跑的补间动画，该动画为水平方向上向右平移 660 像素，持续时间为 3 秒。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:toXDelta="660"
        android:fromYDelta="0"
        android:toYDelta="0"
        android:duration="3000">
    </translate>
</set>
```

② 在 anim 文件夹中创建名称为 rotleft.xml 的 XML 资源文件，在文件中定义一个实现野猪向左奔跑的补间动画，该动画为水平方向上向左平移 660 像素，持续时间为 3 秒。代

码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android" >
    < translate
        android:fromXDelta = "660"
        android:toXDelta = "0"
        android:fromYDelta = "0"
        android:toYDelta = "0"
        android:duration = "3000">
    </translate >
</set >
```

（4）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 控件删除，接着在默认添加的线性布局管理器中添加一个 ImageView 控件，并设置该控件的背景为逐帧动画资源 right.xml，再设置 ImageView 控件的顶外边距和左外边距。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/linearLayout1"
    android:background = "@drawable/bg"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >
    < ImageView
        android:id = "@ + id/imageView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:background = "@anim/right"
        android:layout_marginTop = "280px"
        android:layout_marginLeft = "30px"/>
</LinearLayout >
```

（5）打开默认创建的 MainActivity，在 onCreate()方法中先获取要应用动画效果的 ImageView，并获取向右奔跑和向左奔跑的补间动画资源，然后获取 ImagView 应用的逐帧动画及线性布局管理器，并显示一个消息提示框，再为线性布局管理器添加触摸监听器，在重写 onTouch()方法中，开始播放逐帧动画并播放向右奔跑的补间动画，接下来为向右奔跑和向左奔跑的动画添加动画监听器，并在重写的 onAnimationEnd()方法中改变要使用的逐帧动画和补间动画，播放动画，实现野猪来回奔跑的动画效果。代码为：

```
public class MainActivity extends Activity
{
    private AnimationDrawable anim;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取要应用动画效果的 ImageView
        final ImageView iv = (ImageView)findViewById(R.id.imageView1);
```

```
        //获取向右奔跑的动画资源
        final Animation tright = AnimationUtils.loadAnimation(this, R.anim.rotright);
         //获取向左奔跑的动画资源
        final Animation tleft = AnimationUtils.loadAnimation(this, R.anim.rotleft);
           anim = (AnimationDrawable)iv.getBackground();                //获取应用的帧动画
        //获取线性布局管理器
        LinearLayout ll = (LinearLayout)findViewById(R.id.linearLayout1);
       //显示一个消息提示框
        Toast.makeText(this,"触摸屏幕开始播放...", Toast.LENGTH_SHORT).show();
        ll.setOnTouchListener(new OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                anim.start();                          //开始播放帧动画
                iv.startAnimation(tright);             //播放向右奔跑的动画
                return false;
            }
        });
        tright.setAnimationListener(new AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {}
            @Override
            public void onAnimationRepeat(Animation animation) {}
            @Override
            public void onAnimationEnd(Animation animation) {
               //重新设置 ImageView 应用的帧动画
                iv.setBackgroundResource(R.anim.left);
                iv.startAnimation(tleft);              //播放向左奔跑的动画
                anim = (AnimationDrawable)iv.getBackground();          //获取应用的帧动画
                anim.start();                          //开始播放帧动画
            }
        });
        tleft.setAnimationListener(new AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {}
            @Override
            public void onAnimationRepeat(Animation animation) {}
            @Override
            public void onAnimationEnd(Animation animation) {
                    //重新设置 ImageView 应用的帧动画
                    iv.setBackgroundResource(R.anim.right);
                iv.startAnimation(tright);             //播放向右奔跑的动画
                anim = (AnimationDrawable)iv.getBackground();          //获取应用的帧动画
                anim.start();                          //开始播放帧动画
            }
        });
    }
}
```

(6) 修改动画的标题，打开 res\values 目录下的 string.xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name">奔跑的野猪</string >
    < string name = "action_settings"> Settings </string >
```

```
    <string name="hello_world">Hello world!</string>
</resources>
```

(7) 运行程序,单击屏幕后,屏幕中的野猪将从左侧奔跑到右侧,效果如图 6-19 所示,撞到右侧后,转身向左侧奔跑,直到撞上左侧的栅栏,再转身向右侧奔跑,反复此动画。

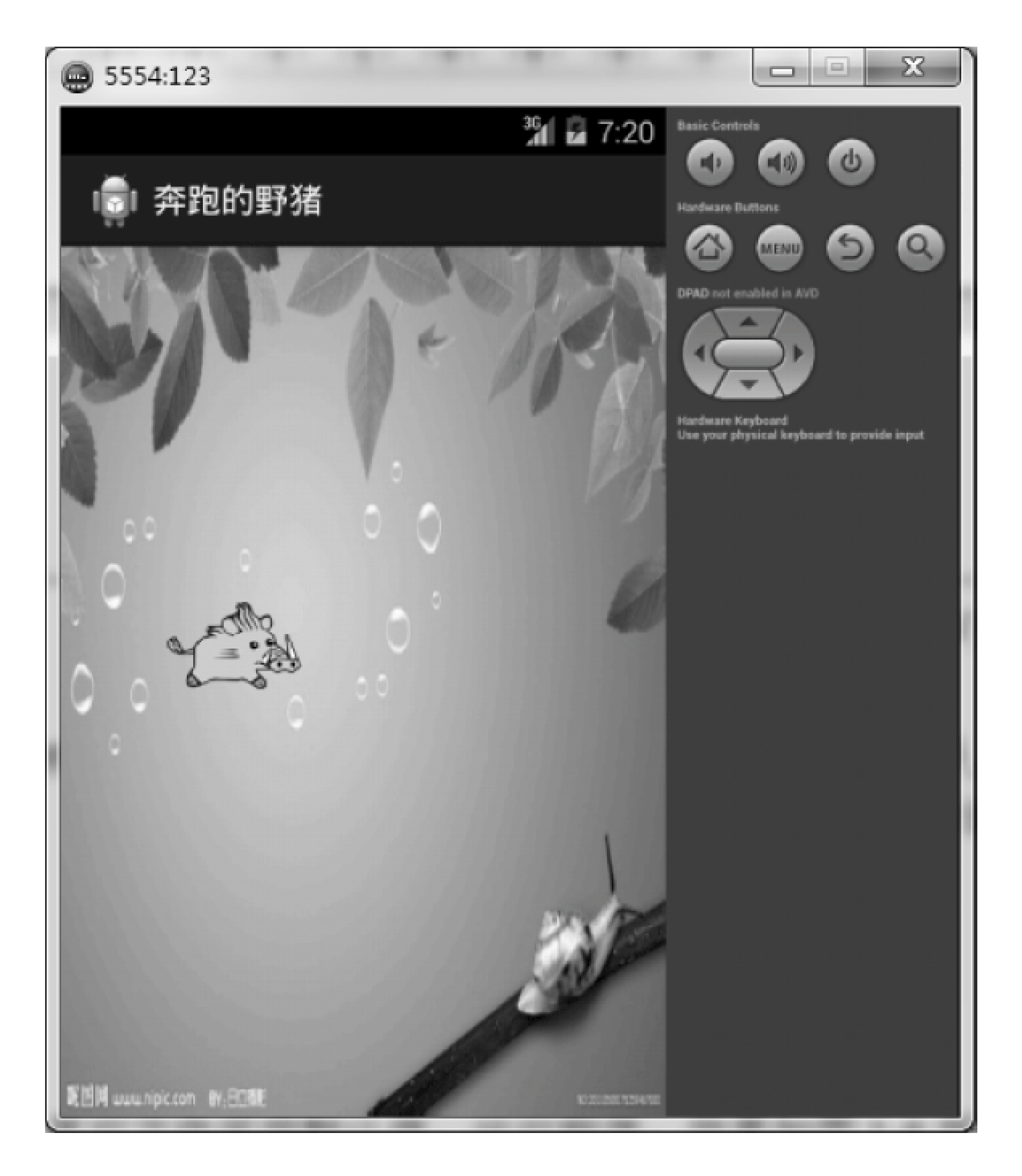


图 6-19　奔跑的野猪

第 7 章 Android 持久化存储

Android 的数据存储方式主要有两种，一种是本机的文本存储，另一种是数据库存储方式。

7.1 文件结构

在学习在 Android 程序中对文件的各种操作之前，先来了解 Android 的文件结构。对于 Android 的文件结构可以分为三大类，分别是系统系统、数据文件和外部存储文件。

7.1.1 系统文件

Android 的系统文件主要存储在\system 文件夹中，下面详细了解\system 文件夹的结构。

- \system\app：该目录文件夹主要存放的是常规下的应用程序，可以看到都是以.apk 为扩展名的 APK 文件，\app 文件夹下的程序为系统默认的组件，安装的软件当然是不会出现在这里的，而是存储在\data 文件夹中。
- \system\bin：该目录文件夹下的文件都是系统的本地程序，即二进制的程序，主要都是 Linux 系统自带的组件(命令)。
- \system\etc：该目录文件夹主要存储着 Android 的系统配置文件，例如 GPS 设置文件(gps.conf)、存储挂载配置文件(mountd.conf)等。
- \system\fonts：该目录文件夹主要存储着与 Android 系统字体相关的文件，例如字体样式、中文字库、unicode 字库等。
- \system\framework：该目录文件夹主要存储着 Android 系统核心文件、系统平台框架核心文件，例如核心库(core.jar)、系统服务(svc.jar)等。
- \system\lib：该目录文件夹主要存储着 Android 系统底层库，例如系统服务组件(libandroid_servers.so)、蓝牙组件(libbluetooth.so)等。
- \system\media：该目录文件夹主要存储着 Android 系统提示事件音和一些常规的铃声，例如闹铃音(alarms)、提示音(notifications)等。
- \system\xbin：该目录文件夹主要存储着 Android 系统的管理工具和配置工具。
- \system\build.prop：该目录文件是一个属性文件，记录着 Android 系统内核、机型、版本以及系统设置和改变等信息。
- \system\usr：该目录文件夹是 Android 用户文件夹，例如共享、键盘布局、时间区域文件等。

- \system\modules：该目录文件夹主要存储着 Android 系统内核模块（主要是 fs 和 net）及模块配置文件。
- \system\lost+found：该目录文件夹是基于 YAFFS（Yet Another Flash Filing System）文件系统固有的，类似回收站的文件夹。
- \system\sd：该目录文件夹是 SD 卡中的 EXT2 分区的挂载目录。

7.1.2 数据文件

用户在使用 Android 系统的过程中会安装应用程序、产生临时数据等，这些用户数据大部分都保存在 data 目录中。

- \data\anr：该目录中保存了\data\anr\traces.txt 文件。当应用程序发生 ANR（Application is Not Responding）错误时，Android 会自动将问题点的 code stack list 写在这个档案内，直接在 Linux 中用 cat 命令查看其内容。
- \data\app：该目录中主要存放的是常规下载的应用程序，可以看到都是以.apk 为扩展名的 APK 文件，与系统文件夹下的不同，在该文件夹中存放的是使用者自己安装的应用程序执行文件（*.apk）。
- \data\app-private：该目录中同样存放的是用户安装的应用程序和执行文件，不过这些文件都是有 DRM 保护的 APK 文件。这类文件较少，该目录一般为空。
- \data\backup：该目录中存放的是备份文件。
- \data\dalvik-cache：该目录中将 apk 中的 DEX 文件安装到 dalvik-cache 目录下（DEX 文件是 Dalvik 虚拟机的可执行文件，其大小约为原始 APK 文件大小的 1/4），当 Android 启动时，DalvikVM 监视所有的程序（APK 文件）和框架，并且为它们创建一个框架和一个依存关系树。DalvikVM 通过这个依存关系树为每个程序优化代码的时候，第一次启动时间非常长的原因就在于一个程序（或者框架库）发生变更，DalvikVM 将会重新优化代码并且再次将其存在缓存中。
- \data\data：该目录中以应用程序名称保存程序的数据。在\data\data\<app-package-name>目录中，在程序中用 Context.openFileOutput()所建立的文件都放在这个目录下的 files 子目录内；而用 Context.getSharedPreferences()所建立的 preferences 文件（*.xml）则是放在 shared_preshared_pref 这个子目录中；建立的数据库文件（*.db）则保存在 databases 子目录中。
- \data\system\：该目录中存放一些配置文件，例如，触摸屏产生的屏校准值就保存在\data\system\calibration 文件中。
- \data\misc：该目录中存放各杂项（功能）所产生的配置文件。

7.1.3 外部存储文件

所谓的外部存储，即 SD 卡储存。对于较大的文件，一般都保存在 SD 卡中。由于该目录简单，在此不再赘述。

对于三大类文件结构，用户在进行操作时要有一定的区别：

- 对于系统文件，如果没有 roots 权限，是无法进行更改甚至读取文件的，所以一般不访问其数据。

- 对于数据文件，应用程序产生的数据默认都保存在其 data\data 目录中，且其他应用程序在没有权限的情况下无法访问，在很大程度上保存了我们的私有数据。

对于外部存储文件，由于其存储容量大，对于保存大文件有优势。但是，对于 SD 卡中的文件，只要应用程序拥有 SD 卡访问权限，就可以访问其中所有的文件，可能存在被其他程序篡改、删除的风险。

7.2 数据存储方式

在掌握了 Android 的文件结构以后，下面来了解 Android 中提供的 5 种数据存储方式。

(1) SharedPreferences 存储数据；

(2) 文件(File)存储数据；

(3) SQLite 数据库存储数据；

(4) ContentProvider 存储数据；

(5) 网络(NetWork)存储数据。

- SharedPreferences 存储：该存储方式适用于简单数据的保存，例如配置属性、保存用户名等具有配置性质的数据保存，但是不适合数据比较大的保存。
- 文件(File)存储：文件存储方式是较常使用的一种保存数据的方式，可以保存较大的数据。而且文件存储不仅能把数据存储在系统中，也能将数据保存到 SD 卡中。
- SQLite 数据库存储：Android 系统提供了 SQLite 标准的数据库、完整支持的 SQL 语句。同样，它可以保存较大的数据，并且可以保存在系统中也可以保存在 SD 卡中。数据库存储具有一定规范的数据非常高效，但是需要数据库的操作规范，相对前两个较复杂。
- ContentProvider 存储：ContentProvider 为存储和获取数据提供了统一的接口，可以在不同的应用程序之间共享数据。
- 网络(NetWork)存储：该存储方式通过网络来获取和存储数据，需要与 Android 网络数据包打交道，与网络相关的应用一般都会使用该存储方式。

7.3 SharedPreferences 存储

7.3.1 SharedPreferences 存储概述

SharedPreferences 是什么样的处理方式呢？其类似于过去 Windows 系统上的 INI 配置文件，但是它分为多种权限，可以全局共享访问，Android123 提示最终是以 XML 方式来保存的，整体效率看来不是特别高，对于常规的轻量级而言比 SQLite 要好很多，如果存储量不大可以考虑自己定义文件格式。XML 处理时 Dalvik 会通过自带底层的本地 XML Parser 解析，比如 XMLpull 方式，这样对于内存资源占用比较好。

它的本质是基于 XML 文件存储 key-value 键值对数据，通常用来存储一些简单的配置信息。

其存储在\data\data\<包名>\shared_prefs 目录下。

SharedPreferences 对象本身只能获取数据；不支持存储和修改，存储修改是通过 Editor 对象实现的。

1. 获取 SharedPreferences 对象

用户可通过以下两种方法来获取 SharedPreferences 的对象。

（1）通过函数 Context. getSharedPreferences(String name，int mode)：其中，name 为本组件的配置文件名(如果想要和本应用程序的其他组件共享此配置文件，可以用这个名字来检索到这个配置文件)；mode 为操作模式，默认的模式为 0 或 MODE_PRIVATE；返回值为 SharedPreferences。

（2）通过函数 Activity. getPreferences(int mode)：其中，配置文件仅可以被调用的 Activity 使用；mode 为操作模式，默认的模式为 0 或 MODE_PRIVATE；返回值为 SharedPreferences。

2. 使用 SharedPreferences 存取数据

保存 key-value 对一般要指定一个文件名，然后用类似 putString 的方法指定 key 和 value。SharedPreferences 也采用了同样的方法。使用 SharedPreferences 保存 key-value 对的步骤如下：

（1）使用 Activity 类的 getSharedPreferences 方法获得 SharedPreferences 对象。其中，存储 key-value 的文件名的名称由 getSharedPreferences 方法的第一个参数指定。

（2）使用 SharedPreferences 接口的 edit 获得 SharedPreferences. Editor 对象。

（3）通过 SharedPreferences. Editor 接口的 putXXX 方法保存 key-value 对。其中，XXX 表示 value 的不同数据类型。Boolean 类型的 value 用 putBoolean 方法，字符串类型的则用 putString 方法。

（4）通过 SharedPreferences. Editor 接口的 commit 方法保存 key-value 对，commit 方法相当于数据库事务中的提交(commit)操作。只有在事件结束后进行提交，才会将数据真正保存到数据库中。保存 key-value 也是一样。

7.3.2 SharedPreferences 存储实例

下面通过一个实例来实现登录效果。

【例 7-1】 使用 SharedPreferences 实现登录界面，并对用户名进行存储。

其实现步骤如下：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li7_1Shared。

（2）打开 res\layout 目录下的 main. xml 布局文件，布局两个编辑框及一个图片按钮控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = "@drawable/h">
< EditText android:layout_width = "185dp"
android:id = "@ + id/user"
android:layout_height = "40dp"
```

```
android:hint = "请输入用户名"
android:singleLine = "true"
android:layout_alignParentTop = "true"
android:layout_alignLeft = "@ + id/pass"
android:layout_marginTop = "66dp">
<requestFocus></requestFocus>
</EditText>
<EditText android:inputType = "textPassword"
android:layout_width = "185dp"
android:id = "@ + id/pass"
android:layout_height = "40dp"
android:hint = "请输入密码"
android:singleLine = "true"
android:layout_below = "@ + id/user"
android:layout_centerHorizontal = "true"
android:layout_marginTop = "44dp">
</EditText>
<ImageButton android:layout_height = "60dp"
android:layout_width = "100dp"
android:id = "@ + id/loginButton"
android:background = "@drawable/ibtn"
android:layout_centerVertical = "true"
android:layout_alignRight = "@ + id/pass"
android:layout_marginRight = "17dp"></ImageButton>
</RelativeLayout>
```

(3) 编写实现登录界面及数据存储的 Activity 文件。打开 src\fs.li7_1shared 下的 MainActivity.java 文件,代码为:

```
package fs.li7_1shared;
import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.EditText;
import android.widget.ImageButton;
public class MainActivity extends Activity
{
    private EditText user = null;
    private EditText password = null;
    private ImageButton loginBtn = null;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        user = (EditText)findViewById(R.id.user);
        password = (EditText)findViewById(R.id.pass);
        loginBtn = (ImageButton)findViewById(R.id.loginButton);
        initView();
        loginBtn.setOnTouchListener(new OnTouchListener())
        {
```

```
            @Override
            public boolean onTouch(View v, MotionEvent event)
            {
                if(event.getAction() == MotionEvent.ACTION_DOWN)
                {
                    v.setBackgroundResource(R.drawable.b1);
                    SharedPreferences userInfo = getSharedPreferences("user_info", 0);
                    userInfo.edit().putString("name", user.getText().toString()).commit();
                    userInfo.edit().putString("pass", password.getText().toString()).commit();
                }
                else if(event.getAction() == MotionEvent.ACTION_UP)
                {
                    v.setBackgroundResource(R.drawable.ibtn);
                }
                return false;
            }
        }};
    }
    private void initView()
    {
        SharedPreferences userInfo = getSharedPreferences("user_info", 0);
        String username = userInfo.getString("name", "");
        String pass = userInfo.getString("pass", "");
        user.setText(username);
        password.setText(pass);
    }
}
```

运行程序，初始界面如图 7-1 所示。单击图中的“请输入用户名”和“请输入密码”两个文本框，即可实现输入功能。



图 7-1　登录初始界面

7.4 文件存储数据

7.4.1 程序私有文件

在 Android 文件结构中，文件分为 3 大类。在进行操作时，一般只操作程序的私有数据和 SD 卡文件。

在应用程序安装到 Android 系统后，这个应用程序的私有文件夹就会被创建，位于 Android 系统的\data\data\<应用程序包名>目录下，默认情况下其他的应用程序都无法在这个文件夹中写入数据。

Android 平台支持 Java 平台下的 I/O 文件操作，实现文件的存储与读取主要使用 FileOutputStream 和 FileInputStream 两个类。

下面通过一个实例来实现 Android 平台下的 I/O 操作。

【例 7-2】 实现 Android 平台下的 I/O 操作。

在实例中，读取数据并显示到屏幕上，其主要功能是在程序启动时创建一个名为 Streamfile.txt 的文件，文件存放在应用程序私有的数据文件夹下，并向文件中写入自定义数据内容，然后读取 Streamfile.txt 文件中的数据内容，并显示到手机屏幕上。

其实现步骤如下：

(1) 在 Eclipse 中创建一个名为 li7_2Stream 的项目。

(2) 打开 res\layout 目录下的 main.xml 文件，布局一个 TextView 控件。代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="@drawable/bj1" >
    <TextView
        android:id="@+id/Text1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1.15" >
    </TextView>
</RelativeLayout>
```

(3) 打开 src\fs.li7_2stream 包下的 MainActivity.java 文件，向其中写入数据，然后读取该文件中的数据内容并显示到界面中。代码为：

```
package fs.li7_2stream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import android.app.Activity;
import android.os.Bundle;
```

```java
import android.widget.TextView;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        String fileName = "Streamfile.txt";               //文件名称
        String content = "欢迎来到 Android";               //指定数据内容
        String result = "";                               //读取文件返回的 String 对象
        boolean istrue = writeFile(fileName, content);    //调用写入数据到文件的方法
        if (istrue) {                                     //判断写入数据是否成功
            result += fileName + "创建成功\n\r";
        } else {
            result += fileName + "创建失败\n\r";
        }
        //调用读取文件的方法,获取返回 String 对象
        result += readFile(fileName);
        TextView textView = (TextView) findViewById(R.id.Text1); //初始化 TextView
        //把读取文件的返回结果显示到 TextView 中
        textView.setText(result);
    }
    /* 向指定的文件中写入指定的数据
     * @param fileName 文件名称
     * @param content 指定数据内容
     * @return boolean 类型(true 表示数据写入成功,false 表示数据写入失败)
     */
    public boolean writeFile(String fileName, String content) {
        try {
            //创建 FileOutputStream 对象,MODE_PRIVATE 为默认模式
            FileOutputStream fOutputStream = openFileOutput(fileName,MODE_PRIVATE);
            //将写入的字符串转换成 byte 数组
            byte[] buffer = content.getBytes();
            fOutputStream.write(buffer);                  //将 byte 数组写入文件
            fOutputStream.flush();                        //清空缓存
            fOutputStream.close();                        //关闭 FileOutputStream 对象
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /* 读取指定文件的数据,并返回 String 对象
     * @param fileName 文件名称
     * @return String 对象
     */
    public String readFile(String fileName) {
        String result = "";                               //返回字符串结果
        try {
            FileInputStream fInputStream = openFileInput(fileName);
                                                          //创建 FileInputStream 对象
            int len = fInputStream.available();           //获取文件的长度
            //创建文件长度大小的 byte 数组
            byte[] buffer = new byte[len];
            fInputStream.read(buffer);                    //将文件流写入 byte 数组
```

```
                //将 byte 数组转换为 String 对象
                result = new String(buffer);
            } catch (Exception e) {
                e.printStackTrace();
            }
            return result;                                  //返回字符串结果
        }
    }
```

运行程序，效果如图 7-2 所示。

在以上代码中，需要重点掌握的是创建文件流的方法，即：



图 7-2　私有文件

```
openFileOutput(String name, int mode)
```

其中，String name 参数是文件名称，int mode 参数是操作模式。该方法为写入数据做准备而打开应用程序私有文件，如果不存在，则在应用程序目录下创建一个文件。其操作模式共有下面 4 种。

- Context.MODE_PRIVATE：值为 0，私有模式，也是默认操作模式，新内容将覆盖原内容。
- Context.MODE_APPEND：值为 32768，追加模式，新内容将追加到原文件后面。
- Context.MODE_WORLD_READABLE：值为 1，允许其他应用程序读取。
- Context.MODE_WORLD_WRITEABLE：值为 2，允许其他应用程序写入，会覆盖原数据。

7.4.2　读/写 SD 卡文件

使用 Activity 的 openFileOutput()方法保存文件，文件是存放在手机空间上的，一般手机的存储空间不是很大，存放一些小文件还行，如果要存放像视频这样的大文件，是不可行的。对于像视频这样的大文件，可以把它存放在 SD 卡中。SD 卡是什么呢？可以把它看作是移动硬盘或 U 盘。

在模拟器中使用 SD 卡，需要先创建一张 SD 卡（当然不是真的 SD 卡，只是镜像文件）。创建 SD 卡可以在 Eclipse 创建模拟器时一起创建，也可以使用 DOS 命令进行创建，例如：

在 DOS 窗口中进入 Android SDK 安装路径的 tools 目录，输入以下命令创建一张容量为 2GB 的 SD 卡，文件扩展名可以任意，建议使用.img：

```
mksdcard 2048M D:\AndroidTool\sdcard.img
```

在程序中访问 SD 卡，需要申请访问 SD 卡的权限。

在 AndroidManifest.xml 中加入访问 SD 卡的权限如下：

```
<!-- 在 SD 卡中创建与删除文件权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<!-- 向 SD 卡写入数据权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

要向 SD 卡存放文件，程序必须先判断手机上是否装有 SD 卡，并且是否可以进行读/写。另外，访问 SD 卡必须在 AndroidManifest.xml 中加入访问 SD 卡的权限，例如：

```
if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED))
{
    File sdCardDir = Environment.getExternalStorageDirectory();  //获取 SD 卡目录
    File saveFile = new File(sdCardDir, "a.txt");
    FileOutputStream outStream = new FileOutputStream(saveFile);
    outStream.write("test".getBytes());
    outStream.close();
}
```

Environment.getExternalStorageState()方法用于获取 SD 卡的状态，如果手机上装有 SD 卡，并且可以进行读/写，那么该方法返回的状态等于 Environment.MEDIA_MOUNTED。

Environment.getExternalStorageDirectory()方法用于获取 SD 卡的目录，当然要获取 SD 卡的目录，也可以这样写：

```
File sdCardDir = new File("/sdcard");                    //获取 SD 卡目录
File saveFile = new File(sdCardDir, "itcast.txt");
```

可以把以上两句代码整合成一句：

```
File saveFile = new File("/sdcard/a.txt");
FileOutputStream outStream = new FileOutputStream(saveFile);
outStream.write("test".getBytes());
outStream.close();
```

下面通过一个实例来演示如何在 SD 卡中读/写文件。

【例 7-3】 实现 SD 卡中文件的操作。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li7_3Scard。

(2) 打开 res\layout 目录下的 main.xml 文件，采用默认代码。

(3) 打开 src\fs.li7_3scard 包下的 MainActivity.java 文件，代码为：

```
package fs.li7_3scard;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        try {
            File sdFile = new File("/sdcard");
            if (sdFile.exists() && sdFile.canWrite()) {
                File txtFile = new File("/sdcard/ghysd.txt");
```

```
                txtFile.createNewFile();
                FileOutputStream fosRef = new FileOutputStream(txtFile);
                fosRef.write("SD 卡的操作".getBytes());
                fosRef.close();
            }
            File readTxtFile = new File("/sdcard/ghysd.txt");
            FileInputStream fisRef = new FileInputStream(readTxtFile);
            InputStreamReader isrRef = new InputStreamReader(fisRef);
            StringBuffer sbRef = new StringBuffer();
            char[] charArray = new char[2];
            int readLength = isrRef.read(charArray);
            while (readLength != -1) {
                sbRef.append(charArray, 0, readLength);
                readLength = isrRef.read(charArray);
            }
            fisRef.close();
            isrRef.close();
            Log.v("----", "" + sbRef.toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

例 7-3 是对手机上的 SD 卡进行操作，下面通过一个实例来演示怎样不通过 SD 卡操作来修改手机中的文件。

【例 7-4】 不通过 SD 卡操作修改手机中的文件。

本例实现：单击“根目录”按钮查询系统的文件名称并显示。在名称界面单击文件名称，如果存在子目录则进入子目录，否则弹出修改文件名称的对话框。在该对话框中输入将要修改的名称，单击“确定”按钮即可改变文件的名称。单击“上翻”按钮，将跳转到本界面的上一界面。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li7_4PhoneFile。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中布局两个线性布局、一个帧布局、两个 Button 控件。代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="@drawable/bj1">
 <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="bottom|right">
```

```
        <Button
            android:id = "@ + id/bgml"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "根目录"
            android:textSize = "18dip"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "bottom|left">
        <Button
            android:id = "@ + id/bsf"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "上翻"
            android:textSize = "18dip"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "360dip"
        android:orientation = "vertical">
        <LinearLayout
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content">
            <TextView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "文件列表:"
                android:textSize = "18dip"
                android:textColor = "#000000"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:paddingTop = "10dip">
            <ListView
                android:id = "@ + id/lvwjlb"
                android:layout_width = "fill_parent"
                android:layout_height = "fill_parent"
                android:textSize = "18dip"
                android:textColor = "#000000"
                android:divider = "#EDAB4A"/>
        </LinearLayout>
    </LinearLayout>
</RelativeLayout>
```

(3) 在 res\layout 目录下创建一个名为 dialog.xml 的文件,在文件中设置 3 个线性布局、一个 TextView 控件、一个 EditText 控件及两个 Button 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
```

```
android:layout_height = "fill_parent"
android:background = "@drawable/bj1">
 < LinearLayout
      android:orientation = "vertical"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent"
      android:paddingLeft = "10dip"
      android:paddingRight = "10dip"
      android:paddingTop = "10dip"
      android:paddingBottom = "10dip"
      android:gravity = "center">
      < LinearLayout
          android:layout_width = "fill_parent"
          android:layout_height = "wrap_content">
          < TextView
              android:layout_width = "fill_parent"
              android:layout_height = "wrap_content"
              android:textSize = "18dip"
              android:textColor = "#ffffff"
              android:paddingLeft = "10dip"
              android:text = "请输入新的文件名称: "/>
      </LinearLayout >
      < LinearLayout
          android:layout_width = "fill_parent"
          android:layout_height = "wrap_content"
          android:paddingLeft = "5dip"
          android:paddingRight = "5dip"
          android:paddingTop = "5dip">
          < EditText
              android:text = ""
              android:textSize = "18dip"
              android:textColor = "#000000"
              android:id = "@+id/et"
              android:layout_width = "fill_parent"
              android:layout_height = "wrap_content"
              android:singleLine = "true"/>
      </LinearLayout >
      < LinearLayout
          android:orientation = "horizontal"
          android:layout_width = "fill_parent"
          android:layout_height = "fill_parent"
          android:gravity = "center">
           < Button
              android:text = "确定"
              android:id = "@+id/bOk"
              android:layout_width = "75dip"
              android:layout_height = "40dip"
              android:textSize = "18dip"
              android:gravity = "center"/>
          < Button
              android:text = "取消"
              android:id = "@+id/bCancle"
              android:layout_width = "75dip"
              android:layout_height = "40dip"
```

```
                android:textSize = "18dip"
                android:gravity = "center"/>
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

（4）打开 src\fs. li7_4phonefile 包下的 MainActivity. java 文件，在该文件中实现手机文件的修改。代码为：

```
public class MainActivity extends Activity
{
    String currentPath;                             //记录当前文件列表的父路径
    String rootPath = "/";                          //根目录
    String leavePath;                               //叶子文件
    Dialog gmDialog;                                //声明改名对话框
    ListView lv;                                    //ListView 控件对象声明
    @Override
    public Dialog onCreateDialog(int id)            //创建对话框
    {
        Dialog result = null;
        switch(id)
        {
            case 0:
                AlertDialog.Builder b = new AlertDialog.Builder(this);
               b.setItems(
                 null,
                 null
               );
               b.setCancelable(false);
               gmDialog = b.create();
               result = gmDialog;
            break;
        }
        return result;
    }
    @Override
    //每次弹出对话框时被回调以动态更新对话框内容的方法
    public void onPrepareDialog(int id, final Dialog dialog)    {
        switch(id)
        {
            case 0:
                dialog.setContentView(R.layout.dialog);
                Button bok = (Button)dialog.findViewById(R.id.bOk);
                Button bcancel = (Button)dialog.findViewById(R.id.bCancle);
                final EditText et = (EditText)dialog.findViewById(R.id.et);
                bok.setOnClickListener                          //确定按钮监听器
                (
                    new OnClickListener()
                    {
                        public void onClick(View arg0)
                        {
                    String newName = et.getText().toString().trim();    //获取新文件的名称
                            //获取修改后的文件路径
                            File xgf = new File(leavePath);
```

```
                    String newPath = xgf.getParentFile().getPath() + "/" + newName;
                    xgf.renameTo(new File(newPath));
                final File[] files = getFiles(currentPath);       //获取根结点文件列表
                  intoListView(files,lv);          //将各个文件添加到 ListView 列表中

                    dialog.cancel();
                  }
                }
            );
            bcancel.setOnClickListener                  //取消按钮监听器
            (
                new OnClickListener()
                {
                  public void onClick(View arg0)
                  {
                    dialog.cancel();
                  }
                }
            );
            dialog.setCancelable(true);
          break;
        }
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    lv = (ListView)MainActivity.this.findViewById(R.id.lvwjlb);     //获取 ListView 控件对象
        Button bgml = (Button)this.findViewById(R.id.bgml);          //搜索根目录文件按钮
        Button bsf = (Button)this.findViewById(R.id.bsf); //搜索父目录文件按钮
        bgml.setOnClickListener                         //搜索根目录文件按钮监听器
        (
            new OnClickListener()
            {
              public void onClick(View v) {
                currentPath = rootPath;
                final File[] files = getFiles(currentPath);       //获取根结点文件列表
                intoListView(files,lv);          //将各个文件添加到 ListView 列表中
              }
            }
        );
        bsf.setOnClickListener                          //搜索父目录文件按钮监听器
        (
            new OnClickListener()
            {
              public void onClick(View v) {
                if((currentPath!= null)&&(!currentPath.equals(rootPath)))
              {//如果当前父路径不是 rootPath,则单击"上翻"按钮,回到上一层目录
              File cf = new File(currentPath);        //获取当前文件列表的路径对应的文件
                    cf = cf.getParentFile();       //获取父目录文件
                    currentPath = cf.getPath();    //记录当前文件列表路径

                    intoListView(getFiles(currentPath),lv);
                }
```

```
                }
            }
        );
    }
    //获取当前目录下的文件列表
    public File[] getFiles(String filePath)
    {
        File[] files = new File(filePath).listFiles();      //获取当前目录下的文件
        return files;
    }
    //将文件列表添加到 ListView 中
    public void intoListView(final File[] files,final ListView lv)
    {
        if(files!= null)                                     //当文件列表不为空时
        {
            if(files.length == 0)
            {       //当前目录为空
                File cf = new File(currentPath);             //获取当前文件列表的路径对应的文件
                cf = cf.getParentFile();                     //获取父目录文件
                currentPath = cf.getPath();                  //记录当前文件列表路径

                Toast.makeText(MainActivity.this, "该文件夹为空!!", Toast.LENGTH_SHORT).
show();
            }
            else
            {
                BaseAdapter ba = new BaseAdapter()//创建适配器
                {
                    public int getCount() {
                        return files.length;
                    }
                    public Object getItem(int position) {
                        return null;
                    }
                    public long getItemId(int position) {
                        return 0;
                    }
                    public View getView(int arg0, View arg1, ViewGroup arg2) {
                        LinearLayout ll = new LinearLayout(MainActivity.this);
                        ll.setOrientation(LinearLayout.VERTICAL);          //竖直排列
                        ll.setPadding(5, 5, 5, 5);                         //留白
                    TextView tv = new TextView(MainActivity.this);         //初始化 TextView
                        tv.setTextColor(Color.BLACK);                      //设置字体颜色
                        tv.setText(files[arg0].getName());                 //添加文件名称
                        tv.setGravity(Gravity.LEFT);        //左对齐
                        tv.setTextSize(18);                 //字体大小
                        ll.addView(tv);                     //LinearLayout 添加 TextView
                        return ll;
                    }
                };
                lv.setAdapter(ba);                           //设置适配器

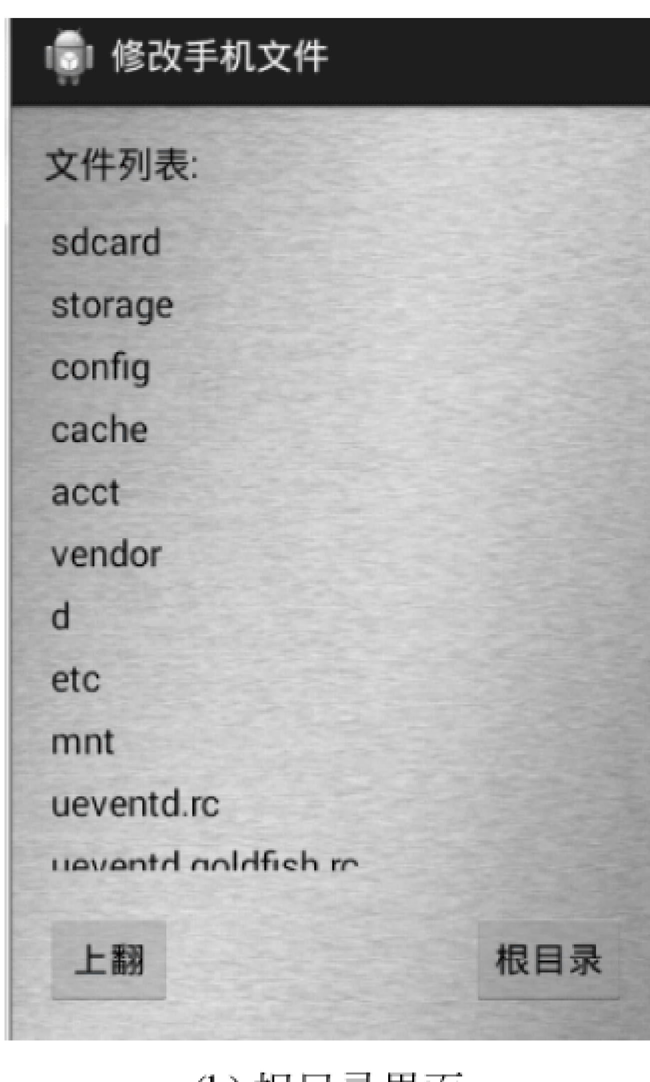
                lv.setOnItemClickListener                    //设置选中菜单的监听器
                (
```

```
                        new OnItemClickListener()
                        {
                          public void onItemClick(AdapterView<?> arg0, View arg1,
                                        int arg2, long arg3) {
                  File f = new File(files[arg2].getPath()); //获得当前单击的文件对象
                                if(f.isDirectory())      //存在分支
                                {
                                    currentPath = files[arg2].getPath();
                      File[] fs = getFiles(currentPath);    //获取当前路径下的所有子文件
                            intoListView(fs,lv);            //将子文件列表填入 ListView 中
                                }
                                else
                                {//弹出对话框,供用户填写新的文件名称
                                    leavePath = f.getPath();
                                    showDialog(0);
                                }
                            }
                        }
                  );
              }
          }
      else
          {
              File cf = new File(currentPath);              //获取当前文件列表的路径对应的文件
              cf = cf.getParentFile();                      //获取父目录文件
              currentPath = cf.getPath();                   //记录当前文件列表路径
              Toast.makeText(MainActivity.this, "该文件夹为空!!", Toast.LENGTH_SHORT).show();
          }
      }
  }
```

运行程序,效果如图 7-3(a)所示。单击界面中的“根目录”按钮,效果如图 7-3(b)所示;单击图 7-3(b)中对应的文件,效果如图 7-3(c)所示。



(a) 默认效果



(b) 根目录界面



(c) 修改文件名称

图 7-3　修改手机文件

例 7-4 实现了对手机文件的修改，下面通过一个实例来演示怎样不通过 SD 卡操作来删除手机中的文件。

【例 7-5】 不通过 SD 卡操作删除手机中的文件。

本例实现：当单击“文件搜索”按钮时，查找 SD 卡中的所有文件，并在该按钮上方的文本框中显示查找的结果；在“文件搜索”按钮下面的文本框中输入要删除的文件名称，单击“删除”按钮，即可实现文件的删除。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li7_5DeleteFile。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中设置 6 个线性布局，声明两个 TextView 控件、一个 ScrollView 控件、两个 EditText 控件和两个 Button 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "@drawable/bj1">
 <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "bottom|right">
        <Button
            android:id = "@ + id/bgml"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "根目录"
            android:textSize = "18dip"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "bottom|left">
        <Button
            android:id = "@ + id/bsf"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "上翻"
            android:textSize = "18dip"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "360dip"
        android:orientation = "vertical">
        <LinearLayout
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content">
            <TextView
```

```
                    android:layout_width = "wrap_content"
                    android:layout_height = "wrap_content"
                    android:text = "文件列表:"
                    android:textSize = "18dip"
                    android:textColor = "#000000"/>
            </LinearLayout >
            < LinearLayout
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:paddingTop = "10dip">
                < ListView
                    android:id = "@+id/lvwjlb"
                    android:layout_width = "fill_parent"
                    android:layout_height = "fill_parent"
                    android:textSize = "18dip"
                    android:textColor = "#000000"
                    android:divider = "#EDAB4A"/>
            </LinearLayout >
        </LinearLayout >
    </RelativeLayout >
```

(3) 打开 src\ fs. li7_5deletefile 包下的 MainActivity. java 文件，在该文件中实现手机文件的删除操作。代码为：

```
package fs.li7_5deletefile;
import java.io.File;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    final EditText etwjlb = (EditText)this.findViewById(R.id.etwjlb);
                                                          //文件列表显示文本框
      final EditText etwjmc = (EditText)this.findViewById(R.id.etwjmc);  //文件名称输入框
        Button bss = (Button)this.findViewById(R.id.bss); //搜索按钮
        Button bsc = (Button)this.findViewById(R.id.bsc); //删除按钮
        bss.setOnClickListener
        (
            new OnClickListener()
            {
                public void onClick(View v) {
                    String f = getFiles();            //获取文件
                    etwjlb.setText(f);                //显示文件
                }
            }
        );
        bsc.setOnClickListener
        (
```

```
                new OnClickListener()
                {
                    public void onClick(View v) {
        File f = new File("/sdcard/" + etwjmc.getText().toString().trim());   //获取当前文件
                        f.delete();                                 //删除该文件
                        String ff = getFiles();                     //获取文件
                        etwjlb.setText(ff);                         //显示文件
                    }
                }
            );
        }
        public String getFiles()
        {
            String result = "";
            File[] files = new File("/sdcard").listFiles();
            if(files == null)
            {
                result = "当前没有文件,无法进行删除操作";
            }
            else
            {
                for(File f:files)
                {
                    result = result + f.getName() + "\n";
                }
            }
            return result;
        }
    }
```

运行程序,效果如图 7-4 所示。

图 7-4　删除手机文件效果图

7.5　SQLite 数据库存储

Android 中的数据库持久化方案采用的是 SQLite,它是一种文件型数据库,支持常用的函数,使用起来比较方便,且文件的体积非常小。

7.5.1　SQLite 数据库存储概述

SQLite 是轻量级嵌入式数据库引擎,它支持 SQL 语言,并且只利用很少的内存就有很好的性能。此外,它还是开源的,任何人都可以使用它。许多开源项目(Mozilla、PHP、Python)都使用了 SQLite. SQLite,它由 SQL 编译器、内核、后端以及附件等组成。SQLite 通过利用虚拟机和虚拟数据库引擎(VDBE)使调试、修改和扩展 SQLite 的内核变得更加方便。

SQLite 数据库存储具有以下特点:

- 面向资源有限的设备;
- 没有服务器进程;
- 所有数据存放在同一文件中,跨平台;

- 可自由复制。

SQLite 内部结构如图 7-5 所示。

SQLite 基本上符合 SQL-92 标准,和其他的主要 SQL 数据库没什么区别。它的优点就是高效,Android 运行时环境包含了完整的 SQLite。

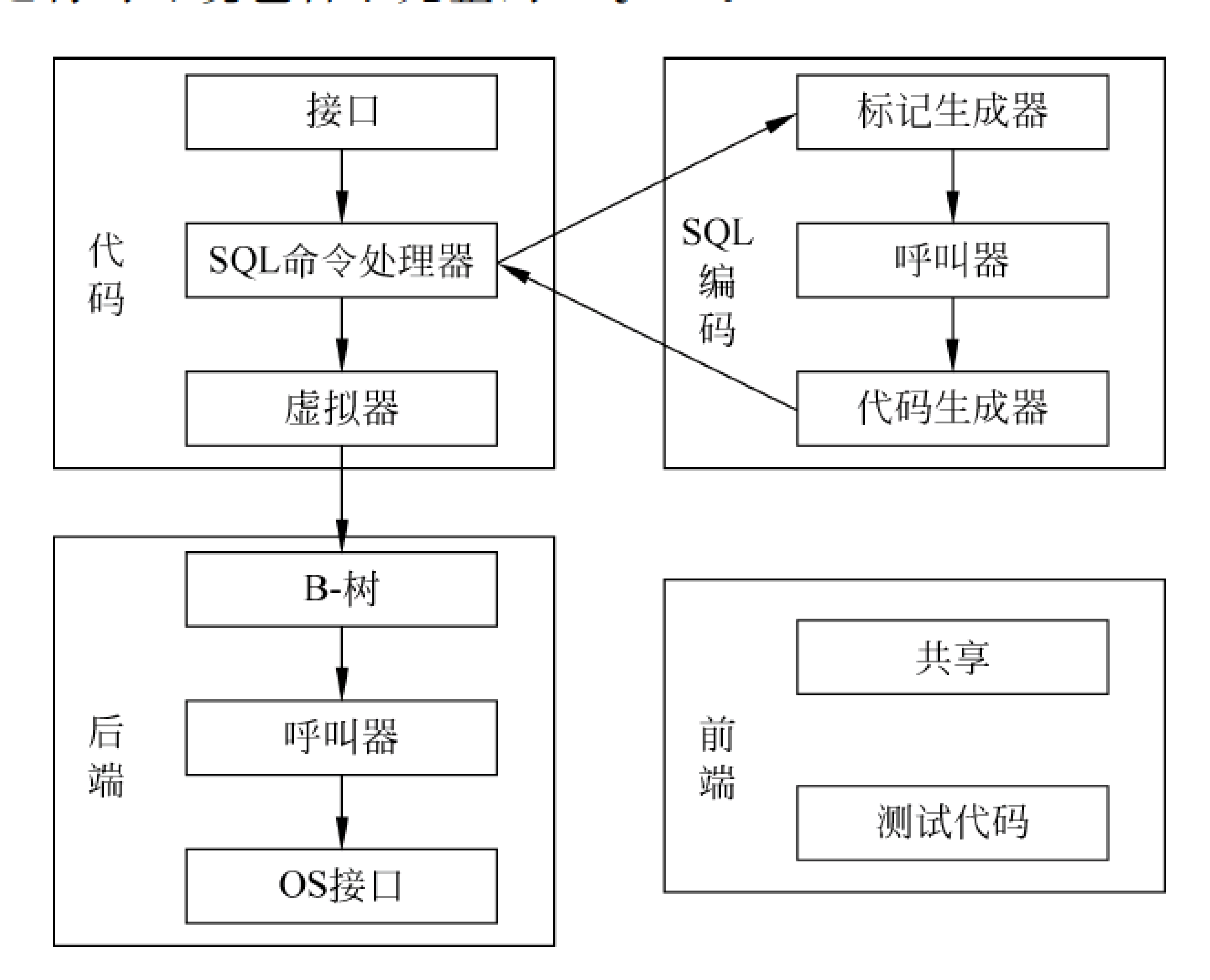


图 7-5　SQLite 内部结构图

SQLite 和其他数据库最大的不同就是对数据类型的支持,在创建一个表时,可以在 CREATE TABLE 语句中指定某列的数据类型,但是用户可以把任何数据类型放入任何列中。当某个值插入数据库时,SQLite 将检查它的类型。如果该类型与关联的列不匹配,则 SQLite 会尝试将该值转换成该列的类型。如果不能转换,则该值将作为其本身具有的类型存储。比如可以把一个字符串(String)放入 INTEGER 列。SQLite 称之为"弱类型"(manifest typing)。此外,SQLite 不支持一些标准的 SQL 功能,特别是外键约束(FOREIGN KEY constrains),嵌套 transaction、RIGHT OUTER JOIN、FULL OUTER JOIN,还有一些 ALTER TABLE 功能。除了上述功能以外,SQLite 还是一个完整的 SQL 系统,拥有完整的触发器、交易等。

Android 集成了 SQLite 数据库,Android 在运行时(run-time)集成了 SQLite,所以每个 Android 应用程序都可以使用 SQLite 数据库。

7.5.2　SQLite 数据库开发

Activites 可以通过 Content Provider 或者 Service 访问一个数据库,下面详细讲解如何创建数据库、添加数据和查询数据库。

1. 创建数据库

Android 不自动提供数据库。在 Android 应用程序中使用 SQLite,必须自己创建数据库,然后创建表、索引,填充数据。Android 提供了 SQLiteOpenHelper 帮助用户创建一个数据库,只要继承 SQLiteOpenHelper 类,就可以轻松地创建数据库。SQLiteOpenHelper 类根据开发应用程序的需要封装了创建和更新数据库使用的逻辑。SQLiteOpenHelper 的子类至少需要实现下面 3 个方法:

- 构造函数，调用父类 SQLiteOpenHelper 的构造函数。这个方法需要 4 个参数，即上下文环境(例如一个 Activity)、数据库名字、一个可选的游标工厂(通常是 Null)、一个代表正在使用的数据库模型版本的整数。
- onCreate()方法，它需要一个 SQLiteDatabase 对象作为参数，根据需要对这个对象填充表和初始化数据。
- onUpgrage()方法，它需要 3 个参数，即一个 SQLiteDatabase 对象、一个旧的版本号和一个新的版本号，这样就可以清楚如何把一个数据库从旧的模型转变到新的模型。

2. 数据库表的基本操作

接下来讨论如何创建表、插入数据、删除表等基本操作。调用 getReadableDatabase()或 getWriteableDatabase()方法，可以得到 SQLiteDatabase 实例，具体调用哪个方法，取决于是否需要改变数据库的内容。例如：

```
b = (new DatabaseHelper(getContext())).getWritableDatabase();
return (db == null) ? false : true;
```

上面这段代码会返回一个 SQLiteDatabase 类的实例，使用这个对象，就可以查询或者修改数据库。当完成了对数据库的操作，需要调用 SQLiteDatabase 的 Close()方法来释放掉数据库连接。

为了创建表和索引，需要调用 SQLiteDatabase 的 execSQL()方法来执行 DDL 语句。如果没有异常，这个方法没有返回值。

例如，可以执行以下代码：

```
db.execSQL("CREATE TABLE ex_table (_id INTEGER PRIMARY KEY
AUTOINCREMENT, title TEXT, value REAL);");
```

这条语句会创建一个名为 ex_table 的表，表有一个列名为_id，并且是主键，该列的值是会自动增长的整数(例如，当插入一行时，SQLite 会给该列自动赋值)，另外还有两列，即 title(字符)和 value(浮点数)。SQLite 会自动为主键列创建索引。

通常情况下，第一次创建数据库时创建了表和索引。如果不需要改变表的 schema，不需要删除表和索引。删除表和索引，需要使用 execSQL()方法调用 DROP INDEX 和 DROP TABLE 语句。

1) 给表添加数据

上面的代码已经创建了数据库和表，现在需要给表添加数据，有两种方法可以给表添加数据。

像上面创建表一样，可以使用 execSQL()方法执行 INSERT(插入)、UPDATE(更新)、DELETE(删除)等语句来更新表的数据。execSQL()方法适用于所有不返回结果的 SQL 语句。例如：

```
db.execSQL("INSERT INTO widgets (name, inventory)" +
"VALUES ('Sprocket', 5)");
```

另一种方法是使用 SQLiteDatabase 对象的 insert()、update()、delete()方法，这些方法把 SQL 语句的一部分作为参数。例如：

```
ContentValues cv = new ContentValues();
cv.put(Constants.TITLE, "example title");
cv.put(Constants.VALUE, SensorManager.GRAVITY_DEATH_STAR_I);
db.insert("ex_table", getNullColumnHack(), cv);
```


update()方法有 4 个参数，分别是表名、表示列名和值的 ContentValues 对象、可选的 WHERE 条件和可选的填充 WHERE 语句的字符串，这些字符串会替换 WHERE 条件中的"?"标记。update()方法根据条件更新指定列的值，所以用 execSQL()方法可以达到同样的目的。

WHERE 条件及其参数和用过的其他 SQL APIs 类似。例如：

```
String[] parms = new String[] {"this is a string"};
db.update("widgets", replacements, "name = ?", parms);
```

delete()方法的使用和 update()类似，使用表名、可选的 WHERE 条件和相应的填充 WHERE 条件的字符串。

2）查询数据库

类似于 INSERT、UPDATE、DELETE，使用 SELECT 有两种方法从 SQLite 数据库检索数据。

（1）使用 rawQuery()直接调用 SELECT 语句。

使用 query()方法构建一个查询。

- Raw Queries

正如 API 的名字，rawQuery()是最简单的解决方法，通过这个方法可以调用 SQL SELECT 语句。例如：

```
Cursor c = db.rawQuery(
  "SELECT name FROM sqlite_master WHERE type = 'table' AND name = 'mytable'", null);
```

以上代码用于检查 table 表是否存在，返回值是一个 cursor 对象，这个对象的方法可以迭代查询结果。

如果查询是动态的，使用这个方法就会非常复杂。例如，当需要查询的列在程序编译的时候不能确定时，使用 query()方法会方便很多。

- Regular Queries

query()方法用 SELECT 语句段构建查询。SELECT 语句内容作为 query()方法的参数，比如要查询的表名、要获取的字段名、WHERE 条件、包含可选的位置参数、去替代 WHERE 条件中位置参数的值、GROUP BY 条件、HAVING 条件。

除了表名，其他参数可以是 null。所以，上面的代码段可以写成：

```
String[] columns = {"ID", "inventory"};
String[] parms = {"snicklefritz"};
Cursor result = db.query("widgets", columns, "name = ?",parms, null, null, null);
```

（2）使用游标。

不管如何执行查询，都会返回一个 Cursor，这是 Android 的 SQLite 数据库游标。使用游标，可以通过下面几种方法：

- 通过使用 getCount()方法得到结果集中有多少记录；
- 通过 moveToFirst()、moveToNext()和 isAfterLast()方法遍历所有记录；
- 通过 getColumnNames()得到字段名；
- 通过 getColumnIndex()转换成字段号；
- 通过 getString()、getInt()等方法得到给定字段当前记录的值；
- 通过 requery()方法重新执行查询得到游标；
- 通过 close()方法释放游标资源。

7.5.3 SQLite 数据库实例

下面通过一个实例实现数据库的基本操作。

【例 7-6】 本例实现对数据库的各种操作，在界面控件上只需进行各种触发按钮的操作，每一个按钮分别实现不同的功能。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li7_63SQLiteOper。

(2) 打开 res\layout 目录下的 main. xml 文件，在该文件中添加"创建数据库"、"创建表"、"SQL 修改数据"、"Android 修改数据"及"查询数据"5 个 Button 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "@drawable/kp">
    <TextView
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
    <Button
        android:id = "@ + id/base"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/table"
        android:layout_below = "@ + id/textView1"
        android:layout_marginTop = "19dp"
        android:text = "创建数据库" />
    <Button
        android:id = "@ + id/table"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/textView1"
        android:layout_below = "@ + id/base"
        android:layout_marginLeft = "18dp"
        android:text = "创建表" />
```

```
    <Button
        android:id = "@ + id/cv_mod"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/table"
        android:layout_below = "@ + id/que"
        android:text = "Android 修改数据" />
    <Button
        android:id = "@ + id/que"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/table"
        android:layout_below = "@ + id/table"
        android:text = "查询数据" />
    <Button
        android:id = "@ + id/sql_mod"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/cv_mod"
        android:layout_below = "@ + id/cv_mod"
        android:text = "SQL 修改数据" />
</RelativeLayout>
```

(3) 创建数据库。打开 src\fs.li7_63 包下的 MainActivity.java 文件,在文件中实现单击"创建数据库"按钮后对数据库的创建。代码为:

```
package fs.li7_63sqliteoper;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.ContentValues;
import android.content.DialogInterface;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
    private Button baseButton;
    private Button tableButton;
    private Button btn_sqlmod, btn_cvmod, btn_qur;
    private final String dbName = "SQLdb";
    private final String tableName = "users";
    private SQLiteDatabase db = null;
    private int i = 1;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.main);
        baseButton = (Button) findViewById(R.id.base); //实例化 Button 对象
        baseButton.setOnClickListener(new OnClickListener()
       { //为 Button 对象添加监听
                    public void onClick(View v) {
                    //创建名为"SQLdb"的数据库
```

```
                db = openOrCreateDatabase(dbName, MODE_PRIVATE, null);
        Toast.makeText(getApplicationContext(), "创建数据库成功", 1000)
                                    .show();
                    }
                };
            }
}
```

运行程序，效果如图 7-6 所示。



图 7-6　单击“创建数据库”按钮效果图

(4) 创建表。在使用 openCreateDatabase 方法之后，会返回一个 android. database. sqlite. SQLiteDatabase 对象，通过该数据库对象即可执行创建表、插入数据等数据库操作。打开 src\fs. li7_63 包下的 MainActivity. java 文件，添加以下代码：

```
tableButton = (Button) findViewById(R.id.table);          //实例化 Button 对象
tableButton.setOnClickListener(new OnClickListener()
{ //为 Button 对象添加监听
            public void onClick(View v) {
                if (db != null) {
                    creatTable();                          //开始创建数据库表
                } else {
                    Toast.makeText(getApplicationContext(), "没有数据库",
                            1000).show();
                }
            }
        });
btn_sqlmod = (Button) findViewById(R.id.sql_mod);
btn_sqlmod.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        //TODO 自动生成的方法存根
        if (db != null) {
            sql_executeData();
        } else {
```

```
                Toast.makeText(getApplicationContext(), "没有数据库", 1000)
                        .show();
            }
        }
    });
```

运行程序,效果如图 7-7 所示。



图 7-7 单击“创建表”按钮的效果图

对于以上代码,需要着重解释以下几点:

- execSQL 是 SQLiteDatabase 对象的一个方法,该方法可以执行一条标准的 SQL 语句,用来创建表,或者操作表中的数据。但是 execSQL 方法的返回值是 void,所以不能进行查询操作。
- query 是 SQLiteDatabase 对象的查询方法。SQLiteDatabase 还提供了 insert、update、delete 等方法来处理数据。
- SQLiteDatabase 的查询操作会返回一个 Cursor 对象,包含了查询出来的各种信息。
- sqlite_master 表是 SQLite 数据库中的管理表,用来定义数据库的模式。这个表是只读的,用户不能对它执行添加、更新或删除操作。sqlite_master 表中的数据会在用户创建或删除表、索引的时候自动更新。

(5) 对表中的数据进行增、删、改操作。

在 Android 应用程序中,要对表中的数据进行添加、删除、修改操作有两种方式,一是通过标准 SQL 语句,另一种是通过 Android 提供的方法。

① SQL 语句。通过 SQLiteDataBase 对象的 execSQL 方法来执行准备好的 SQL 语句。打开 src\fs.li7_63 包下的 MainActivity.java 文件,添加以下代码:

```
btn_sqlmod = (Button) findViewById(R.id.sql_mod);
btn_sqlmod.setOnClickListener(new OnClickListener()
{
    @Override
    public void onClick(View v) {
        //TODO 自动生成的方法存根
        if (db != null) {
            sql_executeData();
        } else {
            Toast.makeText(getApplicationContext(), "没有数据库", 1000)
                    .show();
        }
    }
});
public void sql_executeData() {
String sql;
for (; i < 4; i++) {
    sql = "insert into " + tableName + " values ('" + i
```

```
                + "','a','aabbcc123')";                    //添加一条记录
        db.execSQL(sql);
        }
    sql = "update " + tableName + " set pwd = '0123456' where id = '1'";          //修改一条记录
    db.execSQL(sql);
    sql = "delete from " + tableName + " where id = '2'";      //删除一条记录
    db.execSQL(sql);
    Toast.makeText(getApplicationContext(), "使用 SQL 语句修改数据成功", 1000).show();
        }
```

运行程序,单击"SQL 修改数据"按钮,效果如图 7-8 所示。

当执行上述的数据修改操作后,使用 Android 工具查看表中的内容如下。

```
sqlite> select * from users;
select * from users;
1|a| 333444
3|a| aabbcc123
```

在表中只有两条记录,id 分别为 1 和 3。id 为 1 的记录 pwd 已修改为 333444,id 为 2 的记录已删除,id 为 3 的记录和添加时一致,没有修改。

② Android 方式。除了可以使用标准 SQL 语句外,还可以使用 Android 提供的方法。打开 src\fs. li7_63 包下的 MainActivity. java 文件,添加以下代码:



图 7-8　单击"SQL 修改数据"按钮效果图

```
btn_cvmod = (Button) findViewById(R.id.cv_mod);
btn_cvmod.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        //TODO 自动生成的方法存根
        if (db != null) {
            cv_executeData();
        } else {
            Toast.makeText(getApplicationContext(), "没有数据库", 1000)
                    .show();
        }
    }
});
btn_qur = (Button) findViewById(R.id.que);
btn_qur.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        //TODO 自动生成的方法存根
        if (db != null) {
            queryData();
        } else {
```

```
                    Toast.makeText(getApplicationContext(), "没有数据库", 1000)
                            .show();
                }
            }
        });
    }
    public void cv_executeData() {
        ContentValues cv = new ContentValues();             //实例化 ContentValues 对象
        cv.put("uname", "b");                               //插入字段值
        cv.put("pwd", "abc123");                            //插入字段值
        db.insert(tableName, null, cv);                     //执行 insert 方法
        db.insert(tableName, null, cv);                     //执行 insert 方法
        db.insert(tableName, null, cv);                     //执行 insert 方法
        cv = new ContentValues();                           //实例化 ContentValues 对象
        cv.put("pwd", "654789");                            //插入字段值
        db.update(tableName, cv, "id = ?", new String[] { "4" }); //执行 update 方法
        db.delete(tableName, "id = ?", new String[] { "5" });  //执行 delete 操作
    Toast.makeText(getApplicationContext(), "使用 Android 语句修改数据成功", 1000)
            .show();
        }
```

运行程序，单击“Android 修改数据”按钮，效果如图 7-9 所示。



图 7-9　单击“Android 修改数据”按钮效果图

对于上述代码，需要重点掌握的是数据库类进行数据的添加、修改、删除的方法，分别介绍如下。

- insert(String table, String nullColumnHack, ContentValues values)

insert 参数方法带有 3 个参数，第 1 个参数是要插入数据的表名，字符串形式；第 2 个参数为空字段的名称，字符串形式；第 3 个参数是要插入的数据内容，ContentValues 对象。

ContentValues 是一个 map 形式的集合，用来保存一条记录的字段信息。key 为字段的名称，values 为该字段的值。

- update(String table, ContentValues values, String whereClause, String[] whereArgs)

updage 方法带有 4 个参数，第 1 个参数为要插入数据的表名，字符串形式；第 2 个参数是要更新的数据内容，ContentValues 对象；第 3 个参数是更新条件，字符串形式；第 4 个参数是更新条件的值，字符串数组形式。

更新条件的字符串中的“?”表示一个占位符，其具体的值由第 4 个参数提供。如果直接将参数写入到字符串中，例如“id='5'”的形式，那么第 4 个参数写 null 即可。

- delete(String table, String whereClause, String[] whereArgs)

delete 方法带有 3 个参数，第 1 个参数是要插入数据的表名，字符串形式；第 2 个参数是删除条件，字符串形式；第 3 个参数是要插入数据的值，字符串数组形式。

执行完成表中的记录操作后，使用 Android 工具查看表中的内容如下。

```
sqlite> select * from users;
select * from users;
1|a| 333444
3|a| aabbcc123
4|b| 654789
6|b| abc123
```

在表中有 4 条记录，前两条是使用 SQL 语句进行数据操作后保存的，后两条是使用 Android 的数据库类进行操作后保存的。

(6) 实现查询。在使用 Android 提供的方法时，需要使用的是 query 方法，该方法会返回一个 Cursor 对象，通过对 Cursor 对象的各种操作，可以获得所需的数据。打开 src\fs.li7_63 包下的 MainActivity.java 文件，添加以下代码：

```
btn_qur = (Button) findViewById(R.id.que);
btn_qur.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        //TODO 自动生成的方法存根
        if (db != null) {
            queryData();
        } else {
            Toast.makeText(getApplicationContext(), "没有数据库", 1000)
                    .show();
        }
    }
});
}
public void queryData() {
Cursor cursor = db.query(tableName, null, null, null, null, null, null,
        null); //执行 query 操作获得 Cursor 对象
String str = "";
if (cursor.getCount() != 0) {                          //判断返回的记录条数
    cursor.moveToFirst();                              //游标指向第一条记录
    for (int i = 0; i < cursor.getCount(); i += 1) {
        str = str + cursor.getString(0) + " " + cursor.getString(1)
        + " " + cursor.getString(2) + "\n";            //获取一条记录的每个字段的值
        cursor.moveToNext();                           //游标指向下一条记录
    }
}
//在信息框上显示所有记录信息
new AlertDialog.Builder(MainActivity.this).setTitle("Message")
        .setMessage(str)
        .setNegativeButton("确定", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
            }
            }).show();
        }
}
```

对于上述代码，需要重点讲解的是查询记录和遍历记录的方法。

① 查询方法 query，它有多种重载形式，通常情况下使用 7 个参数的方法：

```
query(String table,String() columns,String selection,String[] selectionArgs,String groupBy,
String having,String orderBy)
```

第 1 个参数为查询表的名字,字符串形式;第 2 个参数为查询的字段名,字符串数组形式;第 3 个参数为查询条件,字符串形式;第 4 个参数是查询条件的值,字符串数组形式;第 5 个参数为分组字段名,字符串形式;第 6 个参数是分组后筛选的条件,字符串形式;第 7 个参数为排序字段名,字符串形式;第 8 个参数为查询结果返回记录条数的限制,字符串形式。

当需要设置查询条件时,参数设置的方式和有条件更新或删除操作是一样的。例如要查询 uname 值为"李凌"的记录,query 方法的参数设置如下:

```
Cursor cursor = db.query(tableName, null, 'uname = ?', new String[] {"李凌"}, null, null, null,
null);
```

如果要对 pwd 字段进行模糊查询,比如查询包含字符"3"的记录,那么 query 方法的参数设置如下:

```
Cursor cursor = db.query(tableName, null, 'pwd like?', new String[] {"%3%"}, null, null, null,
null);
```

由于设置这些参数很麻烦,且拼接 SQL 语句比较烦琐,因此 SQLiteDatabase 对象提供了另一种类似于 java.sql.PreparedStatenment 的查询方式——rawQuery,代码为:

```
String sql = "select * from users where uname = ?And pwd like? ";
Cursor cursor = db.rawQuery(sql,new String[] {"李凌","%3%"});
```

其中,rawQuery 方法有两个参数,第 1 个是 SQL 语句,用占位符表示数值;第 2 个是字符串数组,依次替换 SQL 语句中的占位符。

② 结果遍历 Cursor 对象,其包含了查询结果,并提供了多种方法来操作这些数据。

当 Cursor 对象处理某一条记录时,需要将游标指向该条记录。Cursor 对象提供了 moveToFirst、moveToLast、moveToNext、moveToPrevious 方法将游标指向结果集的第一条、最后一条、下一条、上一条记录,也可以使用 moveToPosition 方法移动到指定位置,该方法需要一个整数为参数。

刚从 query 方法获得 Cursor 对象时,Cursor 对象的游标并非指向第一条记录,而是指向第一条记录的上一行。可以通过 isBeforeFirst 或 isAfterLast 方法判断游标是否指向第一条记录之前或最后一条记录之后,也可以通过 isFirst 或 isLast 方法判断游标是否执行第一条记录或最后一条记录。

Cursor 对象通过一组 getXXX 方法获取一条记录的各个字段的值。这组方法需要一个整数为参数,该整数就是字段的下标,从 0 开始,也可以通过 Cursor 对象的 getColumnCount 方法获取字段的数量。

运行程序,单击"查询数据"按钮,效果如图 7-10 所示。



图 7-10　单击"查询数据"按钮效果

7.6 ContentProvider 存储数据

ContentProvider 为存储和获取数据提供统一的接口，可以在不同的应用程序之间共享数据。Android 已经为常见的一些数据提供了默认的 ContentProvider。

7.6.1 ContentProvider 存储分析

一个程序可以通过实现一个 ContentProvider 的抽象接口将自己的数据完全暴露出来，而且 ContentProviders 是以类似数据库中表的方式将数据暴露，也就是说，ContentProvider 就像一个“数据库”。那么，外界获取其提供的数据，应该和从数据库中获取数据的操作基本一样，只不过是采用 URI 来表示外界需要访问的“数据库”。

Content Provider 提供了一种多应用间数据共享的方式，例如联系人信息可以被多个应用程序访问。

Content Provider 是一个实现了一组用于提供其他应用程序存取数据的标准方法的类。应用程序可以在 Content Provider 中执行查询数据、修改数据、添加数据、删除数据等操作。Android 提供了一些已经在系统中实现的标准 Content Provider，比如联系人信息，图片库等，用户可以用这些 Content Provider 来访问设备上存储的联系人信息、图片等。

1. 查询记录

在 Content Provider 中使用的查询字符串有别于标准的 SQL 查询，诸如 select、add、delete、modify 等操作我们都使用一种特殊的 URI 来进行，这种 URI 由 3 个部分组成，即“content://”、代表数据的路径和一个可选的标识数据的 id。

以下是一些示例 URI：

- content://media/internal/images：这个 URI 将返回设备上存储的所有图片。
- content://contacts/people/：这个 URI 将返回设备上的所有联系人信息。
- content://contacts/people/45：这个 URI 返回单个结果（联系人信息中 id 为 45 的联系人记录）。

尽管这种查询字符串格式很常见，但是它看起来还是有点令人迷惑。为此，Android 提供了一系列的帮助类（在 android.provider 包下），里面包含了很多以类变量形式给出的查询字符串，这种方式更容易让用户理解，参见下例：

```
MediaStore.Images.Media.INTERNAL_CONTENT_URI ontacts.People.CONTENT_URI
```

因此，上面 content://contacts/people/45 这个 URI 就可以写成如下形式：

```
Uri person = ContentUris.withAppendedId(People.CONTENT_URI, 45);
```

然后执行数据查询：

```
Cursor cur = managedQuery(person, null, null, null);
```

这个查询返回一个包含所有数据字段的游标，可以通过迭代这个游标来获取所有的数据。例如，实现依次读取联系人信息表中的指定数据列 name 和 number，代码如下：

```
public class ContentProviderDemo extends Activity
```

```
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
       displayRecords();
    }
    private void displayRecords()
    {
        //该数组中包含了所有要返回的字段
        String columns[] = new String[] { People.NAME, People.NUMBER };
        Uri mContacts = People.CONTENT_URI;
        Cursor cur = managedQuery(
            mContacts,
            columns,                                //要返回的数据字段
            null,                                   //WHERE 子句
            null,                                   //WHERE 子句的参数
            null                                    //Order - by 子句
     );
    if (cur.moveToFirst())
    {
            String name = null;
            String phoneNo = null;
            do{
               //获取字段的值
          name = cur.getString(cur.getColumnIndex(People.NAME));
              phoneNo = cur.getString(cur.getColumnIndex(People.NUMBER));
              Toast.makeText(this, name + " " + phoneNo, Toast.LENGTH_LONG).show();
           } while (cur.moveToNext());
       }
    }
}
```

2. 修改记录

在 Android 中可以使用 ContentResolver.update()方法修改数据,实现代码为:

```
private void updateRecord(int recNo, String name)
{
    Uri uri = ContentUris.withAppendedId(People.CONTENT_URI, recNo);
    ContentValues values = new ContentValues();
    values.put(People.NAME, name);
    getContentResolver().update(uri, values, null, null);
}
```

此时可以调用上面的方法更新指定记录:

```
updateRecord(10, "XYZ");                        //更改第 10 条记录的 name 字段值为"XYZ"
```

3. 添加记录

如果要增加记录,可以调用 ContentResolver.insert()方法,该方法接受一个要增加的记录的目标 URI,以及一个包含了新记录值的 Map 对象,调用后的返回值是新记录的 URI,

包含记录号。

上面的例子都是基于联系人信息簿这个标准的 Content Provider，现在创建一个 insertRecord()方法对联系人信息簿进行数据的添加，实现代码为：

```
private void insertRecords(String name, String phoneNo)
{
    ContentValues values = new ContentValues();
    values.put(People.NAME, name);
    Uri uri = getContentResolver().insert(People.CONTENT_URI, values);
    Log.d("ANDROID", uri.toString());
    Uri numberUri = Uri.withAppendedPath(uri, People.Phones.CONTENT_DIRECTORY);
    values.clear();
    values.put(Contacts.Phones.TYPE, People.Phones.TYPE_MOBILE);
    values.put(People.NUMBER, phoneNo);
    getContentResolver().insert(numberUri, values);
}
```

这样就可以以调用 insertRecords(name, phoneNo)的方式向联系人信息簿中添加联系人姓名和电话号码了。

4. 删除记录

Content Provider 中的 getContextResolver.delete()方法可以用来删除记录。下面的记录用来删除设备上所有的联系人信息：

```
private void deleteRecords()
{
    Uri uri = People.CONTENT_URI;
    getContentResolver().delete(uri, null, null);
}
```

用户也可以指定 WHERE 条件语句来删除特定的记录，实现代码为：

```
getContentResolver().delete(uri, "NAME = " + "'XYZ XYZ'", null);
```

这会删除 name 为“XYZ XYZ”的记录。

7.6.2 ContentProvider 存储实例

本例演示如何对数据库进行简单的操作。

【例 7-7】 本例实现：在主界面中输入姓名，单击“查询”按钮，将查询得到的数据显示在 EditText 文本框中。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用程序，命名为 li7_3ContentP。

(2) 打开 res\layout 目录下的 main.xml 文件，在该文件中创建游标工厂，声明两个 LinearLayout 布局，并在其中一个 LinearLayout 布局中添加 TextView 控件、EditText 控件和 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
```

```
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
        <LinearLayout
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:background = "@drawable/bj1">
          <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "姓名"
            android:textColor = "#FFFFFF"
            android:textSize = "18dip"
            android:paddingRight = "3dip"/>
          <EditText
            android:text = "刘言"
            android:id = "@ + id/EditText1"
            android:layout_width = "150dip"
            android:layout_height = "wrap_content">
          </EditText>
          <Button
              android:id = "@ + id/Button1"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:text = "查询" />
      </LinearLayout>
      <ScrollView
          android:id = "@ + id/ScrollView1"
          android:layout_width = "fill_parent"
         android:layout_height = "wrap_content">
        <EditText
          android:id = "@ + id/EditText2"
          android:layout_width = "fill_parent"
          android:layout_height = "wrap_content">
        </EditText>
      </ScrollView>
   </LinearLayout>
```

(3) 打开 src\fs.li7_3contentp 包下的 MainActivity.java 文件，在该文件中实现按钮的监听，在其中调用 query 方法得到数据，同时遍历所得到的数据，将数据显示在 EditText 文本框中，并将得到的数据添加到文本框中。代码为：

```
package fs.li7_63sqliteoper;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.ContentValues;
import android.content.DialogInterface;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
    private Button baseButton;
    private Button tableButton;
```

```
private Button btn_sqlmod, btn_cvmod, btn_qur;
private final String dbName = "SQLdb";
private final String tableName = "users";
private SQLiteDatabase db = null;
private int i = 1;
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    this.setContentView(R.layout.main);
    baseButton = (Button) findViewById(R.id.base);  //实例化 Button 对象
    baseButton.setOnClickListener(new OnClickListener()
    {//为 Button 对象添加监听
      public void onClick(View v) {
       //创建名为"mydb"的数据库
      db = openOrCreateDatabase(dbName, MODE_PRIVATE, null);
      Toast.makeText(getApplicationContext(), "创建数据库成功", 1000)
                            .show();
                }
            });
    tableButton = (Button) findViewById(R.id.table);//实例化 Button 对象
    tableButton.setOnClickListener(new OnClickListener()
    {//为 Button 对象添加监听
                public void onClick(View v) {
                    if (db != null) {
                        creatTable();               //开始创建数据库表
                    } else {
                        Toast.makeText(getApplicationContext(), "没有数据库",
                                1000).show();
                    }
                }
            });
    btn_sqlmod = (Button) findViewById(R.id.sql_mod);
    btn_sqlmod.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            //TODO 自动生成的方法存根
            if (db != null) {
                sql_executeData();
            } else {
                Toast.makeText(getApplicationContext(), "没有数据库", 1000)
                        .show();
            }
        }
    });
    btn_cvmod = (Button) findViewById(R.id.cv_mod);
    btn_cvmod.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            //TODO 自动生成的方法存根
            if (db != null) {
                cv_executeData();
            } else {
                Toast.makeText(getApplicationContext(), "没有数据库", 1000)
                        .show();
            }
        }
    });
    btn_qur = (Button) findViewById(R.id.que);
```

```
            btn_qur.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    //TODO 自动生成的方法存根
                    if (db != null) {
                        queryData();
                    } else {
                        Toast.makeText(getApplicationContext(), "没有数据库", 1000)
                                .show();
                    }
                }
            });
        }
        public void creatTable() {
    //创建表的 SQL 语句,创建一个名为 users 的表,该表有 id、uname 和 pwd 3 个字段
            String sql = "CREATE TABLE IF NOT EXISTS "
                    + tableName
                    + " (id INTEGER PRIMARY KEY AUTOINCREMENT, uname VARCHAR(50), pwd VARCHAR
    (50));";
            db.execSQL(sql);                                    //执行 SQL 语句
            //查询 sqlite_master 表中类型为 table 的记录的 name 字段
            Cursor cursor = db.query("sqlite_master", new String[] { "name" },
                    "type = ?", new String[] { "table" }, null, null, null, null);
            String tables = "";
            if (cursor.getCount() != 0) {                       //判断查询结果的条数是否为 0
                cursor.moveToFirst();                           //游标指向第一条记录
                for (int i = 0; i < cursor.getCount(); i += 1) {
                    tables = tables + cursor.getString(0) + " "; //累加字符串
                    cursor.moveToNext();                        //游标下移
                }
            }
            //把累加的结果显示在一个信息框中
            new AlertDialog.Builder(MainActivity.this).setTitle("Message")
                    .setMessage(tables)
                    .setNegativeButton("确定", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                        }
                    }).show();
        }
        public void sql_executeData() {
            String sql;
            for (; i < 4; i++) {
                sql = "insert into " + tableName + " values ('" + i
                        + "','a','aabbcc123')";                         //添加一条记录
                db.execSQL(sql);
            }
        sql = "update " + tableName + " set pwd = '333444' where id = '1'";   //修改一条记录
            db.execSQL(sql);
            sql = "delete from " + tableName + " where id = '2'";          //删除一条记录
            db.execSQL(sql);
            Toast.makeText(getApplicationContext(), "使用 SQL 语句修改数据成功", 1000).show();
        }
        public void cv_executeData() {
            ContentValues cv = new ContentValues();             //实例化 ContentValues 对象
            cv.put("uname", "b");                               //插入字段值
```

```
            cv.put("pwd", "abc123");                          //插入字段值
            db.insert(tableName, null, cv);                   //执行 insert 方法
            db.insert(tableName, null, cv);                   //执行 insert 方法
            db.insert(tableName, null, cv);                   //执行 insert 方法
            cv = new ContentValues();                         //实例化 ContentValues 对象
            cv.put("pwd", "654789");                          //插入字段值
            db.update(tableName, cv, "id = ?", new String[] { "4" });   //执行 update 方法
            db.delete(tableName, "id = ?", new String[] { "5" });       //执行 delete 操作
        Toast.makeText(getApplicationContext(), "使用 Android 语句修改数据成功", 1000)
                    .show();
        }
        public void queryData() {
            Cursor cursor = db.query(tableName, null, null, null, null, null, null,
                    null);                                    //执行 query 操作获得 Cursor 对象
            String str = "";
            if (cursor.getCount() != 0) {                     //判断返回的记录条数
                cursor.moveToFirst();                         //游标指向第一条记录
                for (int i = 0; i < cursor.getCount(); i += 1) {
                    str = str + cursor.getString(0) + " " + cursor.getString(1)
                    + " " + cursor.getString(2) + "\n";       //获取一条记录的每个字段的值
                    cursor.moveToNext();                      //游标指向下一条记录
                }
            }
            //在信息框上显示所有记录信息
            new AlertDialog.Builder(MainActivity.this).setTitle("Message")
                    .setMessage(str)
                    .setNegativeButton("确定", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                        }
                    }).show();
        }
}
```

7.7 NetWork 存储数据

前面介绍的几种存储都是将数据存储到本地设备上，除此之外，还有一种存储(获取)数据的方式，即通过 NetWork(网络)来实现数据的存储和获取，可以调用 WebService 返回的数据或是解析 HTTP 协议实现网络数据的交互。

注意： 在 Android 的早期版本中，曾经支持过进行 XMPP Service 和 Web Service 的远程访问。Android SDK 1.0 以后的版本对它以前的 API 做了许多的变更。Android 1.0 以上的版本不再支持 XMPP Service，而且访问 Web Service 的 API 全部变更。

下面通过一个实例来演示 Android 网络存储。

【例 7-8】 本例的功能是通过邮政编码查询美国某个城市的天气预报，以 POST 发送的方式发送请求到 webservicex.net 站点，访问 WebService.webservicex.net 站点上提供查询天气预报的服务。具体请参考其 WSDL 文档，网址如下：

```
http://www.webservicex.net/WeatherForecast.asmx?WSDL
```

输入：美国某个城市的邮政编码。

输出：该邮政编码对应城市的天气预报。

实现该实例的代码为：

```
package fs.li7_8Net;
import java.util.ArrayList;
import java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import android.app.Activity;
import android.os.Bundle;
public class MyAndroidWeatherActivity extends Activity
{
    //定义需要获取的内容来源地址
    private static final String SERVER_URL =  "http://www.webservicex.net/WeatherForecast.
asmx/GetWeatherByPlaceName";
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //根据内容来源地址创建一个 HTTP 请求
        HttpPost request = new HttpPost(SERVER_URL);
        //添加一个变量
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        //设置一个地区名称
    params.add(new BasicNameValuePair("PlaceName", "NewYork"));            //添加必需的参数
        try {
            //设置参数的编码
            request.setEntity(new UrlEncodedFormEntity(params, HTTP.UTF_8));
            //发送请求并获取反馈
            HttpResponse httpResponse = new DefaultHttpClient().execute(request);
            //解析返回的内容
            if(httpResponse.getStatusLine().getStatusCode() != 404)
            {
                String result = EntityUtils.toString(httpResponse.getEntity());
                System.out.println(result);
            }
        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

在配置文件中设置访问网络权限如下：

```
<uses-permission android:name="android.permission.INTERNET" />
```

以上代码使用 HTTP 从 webservicex 获取 ZipCode 为"200120"(美国 WASHINGTON D. C)的内容，其返回的内容如下：

```
<WeatherForecasts xmlns:xsd = "http://www.w3.org/2001/ XMLSchema" xmlns:xsi = "http: //www.
w3.org/2001/XMLSchema- instance" xmlns = "http://www.webservicex.net">
 <Latitude>38.97571</Latitude>
 <Longitude>77.02825</Longitude>
 <AllocationFactor>0.024849</AllocationFactor>
 <FipsCode>11</FipsCode>
 <PlaceName>WASHINGTON</PlaceName>
 <StateCode>DC</StateCode>
 <Details>
    <WeatherData>
      <Day>Saturday, April 25, 2009</Day>
      <WeatherImage>http://forecast.weather.gov/ images/wtf/sct.jpg</WeatherImage>
      <MaxTemperatureF>88</MaxTemperatureF>
      <MinTemperatureF>57</MinTemperatureF>
      <MaxTemperatureC>31</MaxTemperatureC>
      <MinTemperatureC>14</MinTemperatureC>
    </WeatherData>
    <WeatherData>
      <Day>Sunday, April 26, 2009</Day>
      <WeatherImage>http://forecast.weather.gov/ images/wtf/few.jpg</WeatherImage>
      <MaxTemperatureF>89</MaxTemperatureF>
      <MinTemperatureF>60</MinTemperatureF>
      <MaxTemperatureC>32</MaxTemperatureC>
      <MinTemperatureC>16</MinTemperatureC>
    </WeatherData>
…
 </Details>
</WeatherForecasts>
```

这个例子演示了如何在 Android 中通过网络获取数据。

第 8 章　Android 手机通信与服务

网络通信是交换网络数据的手段，它具有浏览网页、收发电子邮件、进行视频通话和电视直播等功能。不只在 PC 上网络通信必不可少，在手机中，网络通信也是一个重要的功能。在 Android 系统中，人们同样可以通过网络通信随时随地浏览网页、即时聊天、收发微博等。

Android 手机具有强大的自动服务功能，其自动服务功能包含应用程序的后台运行等。

8.1　RPC 通信

Android 系统中的进程间通信是通过一个轻量级的 RPC（Remote Procedure Call，远程进程调用）和 AIDL（Android Interface Definination Language）规范来生成两个进程之间可以相互访问的代码。其中，RPC 是以接口方式来实现的，客户端与被调用实现之间是通过代理模式来实现的，它们又是以 Java 的 RMI 和代理模式为理论基础的。

有关代理模式的知识，可以用图 8-1 和图 8-2 这两个思维导图来表示。

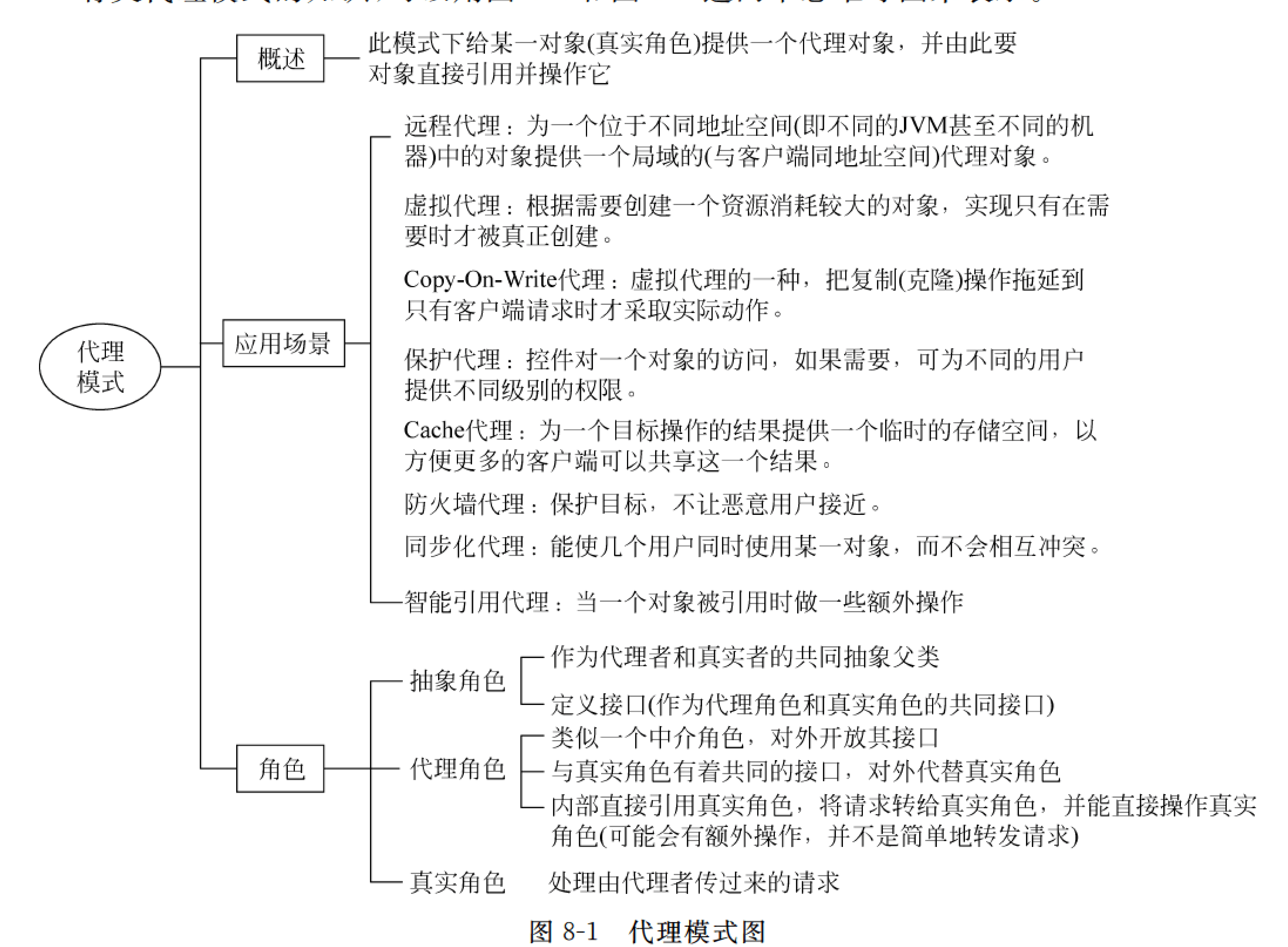


图 8-1　代理模式图

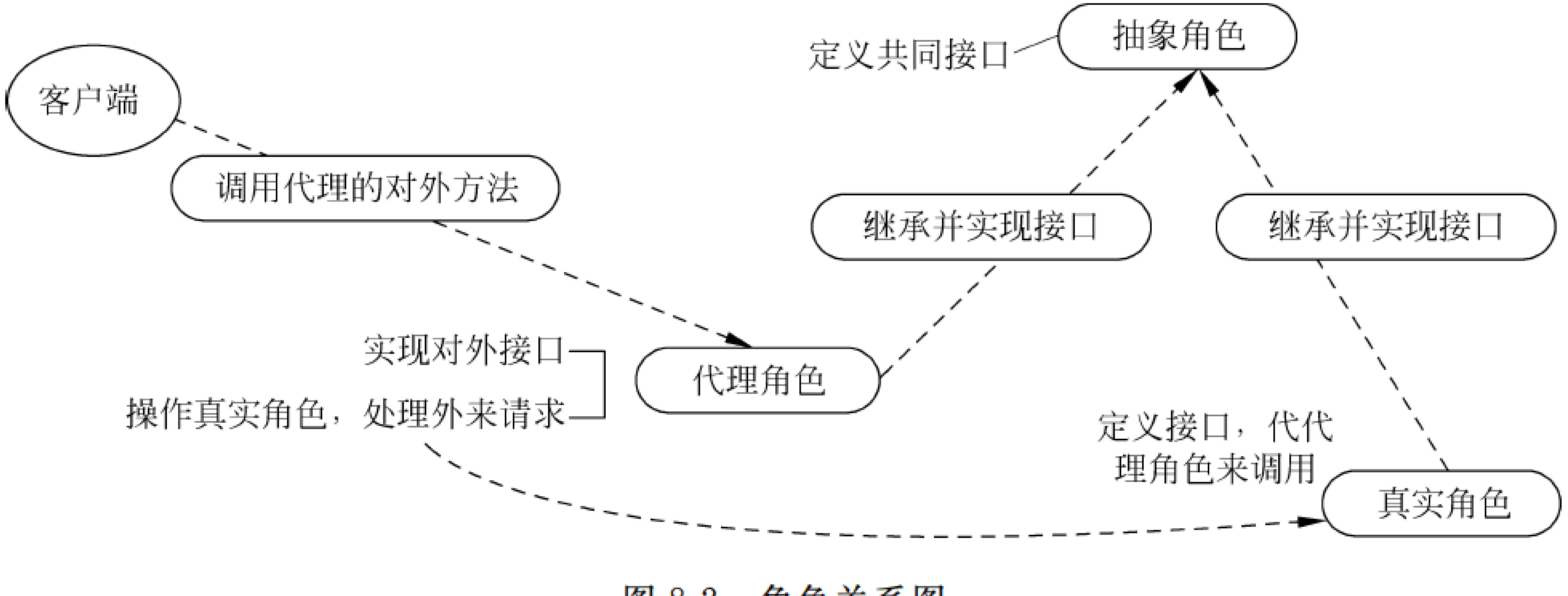


图 8-2　角色关系图

这里以一个代码实例来说明实际运用。

(1) 抽象类 Role 代码：

```
package fs.proxy;
//代理角色和真实角色的共同抽象类
public abstract class Role
{                                              //作为代理角色和真实角色的共同接口,方便
                                               //代理角色对外代替真实角色提供服务
   public abstract void service(String user_id);
}
```

(2) 真实角色类 RealRole 代码：

```
package fs.proxy;
/*
 * 真实角色类
 * 对外不可访问
 */
public class RealRole extends Role {
    /* (non-Javadoc)
     * 提供服务
     */
    @Override
    public void service(String user_id) {
        System.out.println("真实角色为你服务…");
    }
    //验证用户身份
    public boolean CheckUser(String user_id)
    {
        return true;
     }
   }
```

(3) 代理类 ProxyRole 代码：

```
package fs.proxy;
/*
 * 代理角色类
 * 对客户端开发其接口
```

```
 * 内部可以直接引用真实角色实例,将客户端的请求转给真实角色实例
 * 在转发请求的前或者后面可以增加一些额外操作
 */
public class ProxyRole extends Role {
    private RealRole realrole = null;
    /* (non-Javadoc)
     * @see com.magc.proxy.Role#service()
     */
    @Override
    public void service(String user_id) {
        System.out.println("代理角色为你服务…");
        //需要时才去创建真实角色实例
        realrole = new RealRole();
        //增加额外操作: 验证身份
        System.out.println("验证身份…");
        if(!realrole.CheckUser(user_id))
            return;
        System.out.println("去找真实角色实例帮忙处理事务…");
        realrole.service("magc");
        System.out.println("欢迎光临…");
    }
}
```

(4) 测试类 RoleTest 代码:

```
package fs.proxy;
/*
 * 代理模式测试类
 * 作为客户端去请求调用代理类的接口
 * 客户端只能访问代理类,不能访问真实角色类
 */
public class ProxyTest {
   public static void main(String[] args) {
   ProxyRole proxy = new ProxyRole();
   proxy.service("magc");
   }
}
```

8.2 TCP 通信

本节使用面向连接的 Socket 通信。所谓的 Socket 通常也称作“套接字”,应用程序通常通过“套接字”向网络发出请求或者应答网络请求。

根据连接启动的方式以及本地套接字要连接的目标,套接字之间的连接过程可以分为 3 个步骤,即服务器监听、客户端请求、连接确认。

- 服务器监听:服务器端套接字并不定位具体的客户端套接字,而是处于等待连接的状态,实时监控网络状态。
- 客户端请求:由客户端的套接字提出连接请求,要连接的目标是服务器端的套接字。为此,客户端的套接字必须首先描述它要连接的服务器端的套接字,指出服务器端的套接字的地址和端口号,然后向服务器端套接字提出连接请求。

- 连接确认：指当服务器端套接字监听到或者接收到客户端套接字的连接请求时，它就响应客户端套接字的请求，建立一个新的线程，把服务器端套接字的描述发给客户端，一旦客户端确认了此描述，连接就建立好了。而服务器端套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。

Socket 必须在发送数据之前和目的地的 Socket 建立好连接。所以，该模式下的通信，服务器端先启动侦听服务，等待客户端的连接。客户端连接到服务端后发送请求到服务器端，服务器端处理请求并做出相应的应答，实现通信，流程如图 8-3 所示。

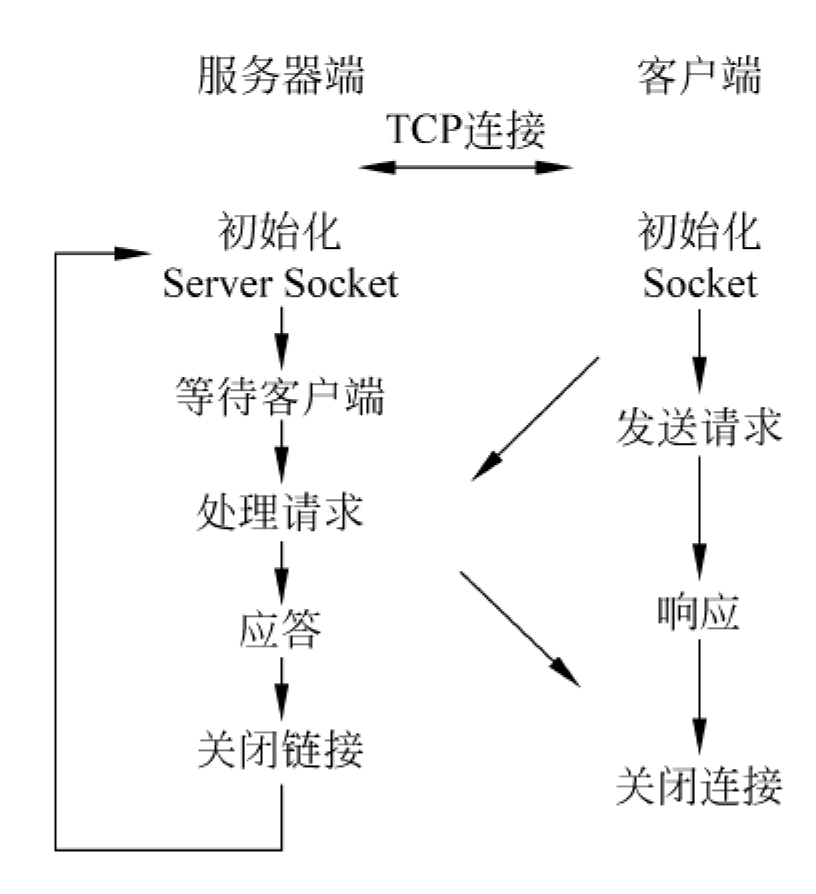


图 8-3　TCP 通信流程图

8.2.1　TCP 通信概述

PC 的 IP 相对固定，作为服务器端运行，需要完成服务器端的 TCP 通信流程以及关闭 PC 的操作。值得注意的是，由于服务器端是运行在 PC 的程序，所以需要创建的是一个 Java 的标准项目，而不是 Android 项目。

从 TCP 的通信流程可以看出，在服务器端需要完成以下 4 个步骤。

（1）创建服务器端套接字并绑定到一个端口。在 Java 标准接口中提供了 ServerSocket 和 Socket 两个类，分别用来表示服务器端和客户端。服务器端的 ServerSocket 有以下几种构造函数。

ServerSocket()

ServerSocket(int aport)

ServerSocket(int aport,int backlog)

ServerSocket(int aport,int backlog,InetAddress localAddr)

其中，参数 aport 指定服务器要绑定的端口即为服务器要监听的端口，参数 backlog 指定客户连接请求队列的长度，参数 logcalAddr 指定服务器要绑定的 IP 地址。一般来说，端口号 0 到 923 为系统预留的，使用的端口最好大于 924。例如创建一个监听端口号为 3344 的服务套接字，代码为：

```
ServerSocket serversocket = new ServerSocket(3344);
```

（2）套接字设置监听模式等待连接请求。在创建服务套接字后，接下来就要监听端口等待客户端的连接，使用 ServerSocket 类的方法为：

```
Socket accept()
```

该方法为一个阻塞方法，调用该方法后将一直监听端口等待客户端的请求，直到有客户端连接到该端口，才会返回一个对应客户端的 Socket，继续执行之后的代码。

（3）接受连接请求后进行通信。Socket 连接建立后，服务器端和客户端通过 Socket 的输入流、输出流来读\写数据，实现通信的功能。Socket 提供的方法为：

```
InputStream getInputStream()
OutputStream getOutputStream()
```

它们分别返回用于读取数据的 InputStream 类对象和用于写入数据的 outputStream 类对象。为了方便读/写数据,可以使用 DataInputStream 和 DataOutputStream 类;对于文本流对象,可以使用 InputStreamReader 和 OutputStreamReader 类。在此以使用 DataInputStream 类读取输入请求为例,代码为:

```
DataInputStream data_input = new DataInputStream(Client_socket,getInputStream());
String msg = data_input,readUTF();
```

(4) 关闭该 Socket 返回,等待下一个连接请求。通信完成后,需要将输入流、输出流以及 Socket 关闭,以主动释放不再使用的资源。

8.2.2 TCP 通信实例

在熟悉了整个通信过程以及关键点后,实现在 PC 上运行的服务器端,对客户端输入的命令进行判断,执行不同命令对应的操作。

【例 8-1】 实现 Android 客户端 Socket 连接 PC 服务器端。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li8_1Socket_PC。

(2) 打开 res\layout 目录下的 main. xml 文件,在该文件中布局一个 TextView 控件、一个 EditText 控件和一个 Button 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
     android:background = "@drawable/kp" >
     <TextView
         android:text = "TCP 通信"
         android:id = "@ + id/TextView01"
         android:layout_width = "wrap_content"
         android:layout_height = "wrap_content"/>
     <EditText
         android:id = "@ + id/EditText01"
         android:layout_width = "wrap_content"
         android:layout_height = "wrap_content"
         android:layout_alignLeft = "@ + id/Button01"
         android:layout_below = "@ + id/Button01"
         android:layout_marginTop = "36dp"
         android:ems = "10"
         android:text = "输入流"/>
      <Button
         android:id = "@ + id/Button01"
         android:layout_width = "wrap_content"
         android:layout_height = "wrap_content"
         android:layout_alignLeft = "@ + id/TextView01"
         android:layout_below = "@ + id/TextView01"
```

```
        android:layout_marginTop = "20dp"
        android:text = "关闭" />
</LinearLayout>
```

(3) 打开 src\fs. li8_1socket_pc 包下的 MainActivity. java 文件，在该文件中创建 Socket 及实现 Socket 连接。代码为：

```
package fs.li8_1socket_pc;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends Activity{
    private TextView mTextView = null;
    private EditText mEditText = null;
    private Button mButton = null;
    /* 第一次调用 Activity */
        @Override
        public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mButton = (Button) findViewById(R.id.Button01);
        mTextView = (TextView) findViewById(R.id.TextView01);
        mEditText = (EditText) findViewById(R.id.EditText01);
      // 登录
     mButton.setOnClickListener(new OnClickListener(){
         public void onClick(View v){
             Socket socket = null;
             String message = mEditText.getText().toString() + "\r\n";
               try {
                   //创建 Socket
                    socket = new Socket("192.168.1.100",5554);
                    // 查看本机 IP,每次开机都不同
                    // 向服务器发送消息
                            PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())),true);
                    out.println(message);
                   // 接收来自服务器的消息
                    BufferedReader br = new BufferedReader(new InputStreamReader(socket.
getInputStream()));
                    String msg = br.readLine();
                    if (msg != null) {
                        mTextView.setText(msg);
                    } else {
```

```
                    mTextView.setText("数据错误!");
                }
                //关闭流
                out.close();
                br.close();
                // 关闭 Socket
                socket.close();
            } catch (Exception e){
                // TODO  异常处理
                Log.e("", e.toString());
            }
        }
    });
  }
}
```

(4) 在 src\fs.li8_1socket_pc 包下创建一个 Server.java 文件,用于实现服务器端连接。代码为:

```
package fs.li8_1socket_pc;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
public class Server implements Runnable {
    public void run() {
        try {
            //创建 ServerSocket
         ServerSocket serverSocket = new ServerSocket(5554);
            while (true) {
                //接受客户端请求
                Socket client = serverSocket.accept();
                System.out.println("accept");
                try {
                    //接收客户端消息
                    BufferedReader in = new BufferedReader(
                            new InputStreamReader(client.getInputStream()));
                    String str = in.readLine();
                    System.out.println("read:" + str);
                    //向服务器发送消息
                    PrintWriter out = new PrintWriter(new BufferedWriter(
                            new OutputStreamWriter(client.getOutputStream())),true);
                    out.println("server message");
                   //关闭流
                    out.close();
                    in.close();
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                    e.printStackTrace();
                } finally {
                    //关闭
```

```
                        client.close();
                        System.out.println("close");
                    }
                }
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
        // main 函数,开启服务器
        public static void main(String a[]) {
            Thread desktopServerThread = new Thread(new Server());
            desktopServerThread.start();
        }
}
```

(5) 打开 AndriodManifest.xml 文件,将代码修改为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.li8_1socket_pc"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.li8_1socket_pc.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
   < uses - permission android:name = "android.permission.INTERNET"/>
</manifest >
```

运行程序,效果如图 8-4 所示。



图 8-4　TCP 通信

【例 8-2】 Android 客户端连接 PC 服务器端(Socket 连接)。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li8_2PC_Socket。

(2) 编写 res\layout 目录下的 main.xml 文件的代码,与例 8-1 类似。

(3) 打开 src\fs.li8_2pc_socket 包下的 MainActivity.

java 文件,实现客户端连接。代码为:

```
package fs.li8_2pc_socket;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.view.View;
import android.view.View.OnClickListener;
public class MainActivity extends Activity implements OnClickListener{
    Socket socket;
    DataInputStream dis;
    DataOutputStream dos;
    private TextView mTextView1;
    private Button Button01;
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mTextView1 = (TextView) findViewById(R.id.rec);
        Button01 = (Button) findViewById(R.id.Button01);
        Button01.setOnClickListener(this);
        Client("127.0.0.1");
        //WriteInt(25);
        //WriteString("client");
        //ReadInt();
        }
    public void Client(String IP) {
        try {
            // 创建一个 Socket 流连接到目标主机
            socket = new Socket(IP, 10000);
            // 输入流读出数据,输出流写数据
            dis = new DataInputStream(socket.getInputStream());
            dos = new DataOutputStream(socket.getOutputStream());
            } catch (IOException ioe) {
                ioe.printStackTrace();
                }
        }
    // 写数据到 Socket
    public void WriteInt(int i) {
        try {
            dos.writeInt(i);
            dos.flush();
            } catch (IOException ioe) {
                ioe.printStackTrace();
                }
        }
    public void WriteString(String str){
        StringBuffer message = new StringBuffer();
```

```
            message.append(str);
            byte[] b = new byte[6];
            try {
                dos.write(12);
                } catch (IOException e) {
                    // TODO 自动生成的方法存根
                    e.printStackTrace();
                    }
            }
    // 显示从 Socket 返回的数据
    public void ReadInt() {
            try {
                mTextView1.setText(dis.readInt());
                System.out.println(dis.readInt());
                } catch (IOException ioe) {
                    ioe.printStackTrace();
                    }
            }
    @Override
    public void onClick(View v) {
        // TODO 自动生成的方法存根
        WriteInt(25);
        }
    }
```

(4) 在 src\fs.li8_2pc_socket 包下新建一个 Server.java 文件,实现服务端。代码为:

```
package fs.li8_2pc_socket;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class Server {
    ServerSocket serversocket;
    Socket socket;
    DataInputStream dis;
    DataOutputStream dos;
    public Server() {
        try {
            serversocket = new ServerSocket(1900);
            System.out.println("等待 Client 连接");
            // 侦听套接字上的连接
            socket = serversocket.accept();
            System.out.println("Client 已连接");
            dis = new DataInputStream(socket.getInputStream());
            dos = new DataOutputStream(socket.getOutputStream());
            } catch (IOException ioe) {
                ioe.printStackTrace();
                }
            }
            public void WriteInt(int i){
                try {
                    //向 Socket 的输出流写入 Int(32 位 integer)数据
```

```
                    // 如果是 4 个字节,则从高到低写入
                    dos.writeInt(i);
                    dos.flush();
                    } catch (IOException ioe) {
                        ioe.printStackTrace();
                        }
                    }
                    //从 Socket 的输入流读出 int(32 位 integer)数据
                    public void ReadInt() {
                        try {
                            System.out.println(dis.readInt());
                            } catch (IOException ioe) {
                                ioe.printStackTrace();
                                }
                        }
                        public void readString(){
                            try{
                                System.out.println(dis.readUTF());
                                }catch(IOException e){
                                    e.printStackTrace();
                                    }
                            }
                        public static void main(String args[]){
                        Server theServer = new Server();
                        // 接收服务器端已经成功获取客户端发送来的数据
                        theServer.ReadInt();
                        theServer.readString();
                        theServer.WriteInt(10);
                }
        }
```

(5) AndroidManifest.xml 文件的代码不需要改变。

运行程序,效果与图 8-4 类似。

8.3 UDP 通信

UDP 通信方式是无连接的 Socket 通信,所以不需要像 TCP 那样先建立连接再发送数据,可以直接对目标地址发送数据。这种方式更加快速、高效,但是不能保证数据能够完全到达目标端。

8.3.1 UDP 通信概述

UDP(User Datagram Protocol,用户数据报协议)是 OSI(Open System Interconnection,开放式系统互连)参考模型中的一种无连接的传输层协议,提供面向事务的简单不可靠信息传送服务。IETF RFC 768 是 UDP 的正式规范,UDP 在 IP 报文中的协议号是 17。

在网络中,UDP 和 TCP 协议一样用于处理数据包,是一种无连接的协议。在 OSI 模型中,它在第四层——传输层,处于 IP 协议的上一层。UDP 有不提供数据包分组、组装和不能对数据包进行排序的缺点,也就是说,当报文发送之后是无法得知其是否安全、完整的到达的。UDP 用来支持那些需要在计算机之间传输数据的网络应用,包括网络视频会议系统

在内的众多的客户/服务器模式的网络应用都需要使用 UDP 协议。UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩被一些类似协议所掩盖，但是即使是在今天，UDP 仍然不失为一项非常实用和可行的网络传输层协议。

与人们所熟知的 TCP(传输控制协议)协议一样，UDP 协议直接位于 IP(网际协议)协议的顶层。根据 OSI(开放式系统互连)参考模型，UDP 和 TCP 都属于传输层协议。

UDP 协议的主要作用是将网络数据流量压缩成数据包的形式。一个典型的数据包就是一个二进制数据的传输单位。每一个数据包的前 8 个字节用来包含报头信息，剩余字节则用来包含具体的传输数据。

8.3.2 UDP 通信流程

UDP 通信相对于 TCP 通信而言比较简单，不需要事先建立连接，只需要创建一个接收和发送的套接字即可实现数据处理和发送，UDP 的通信流程如图 8-5 所示。

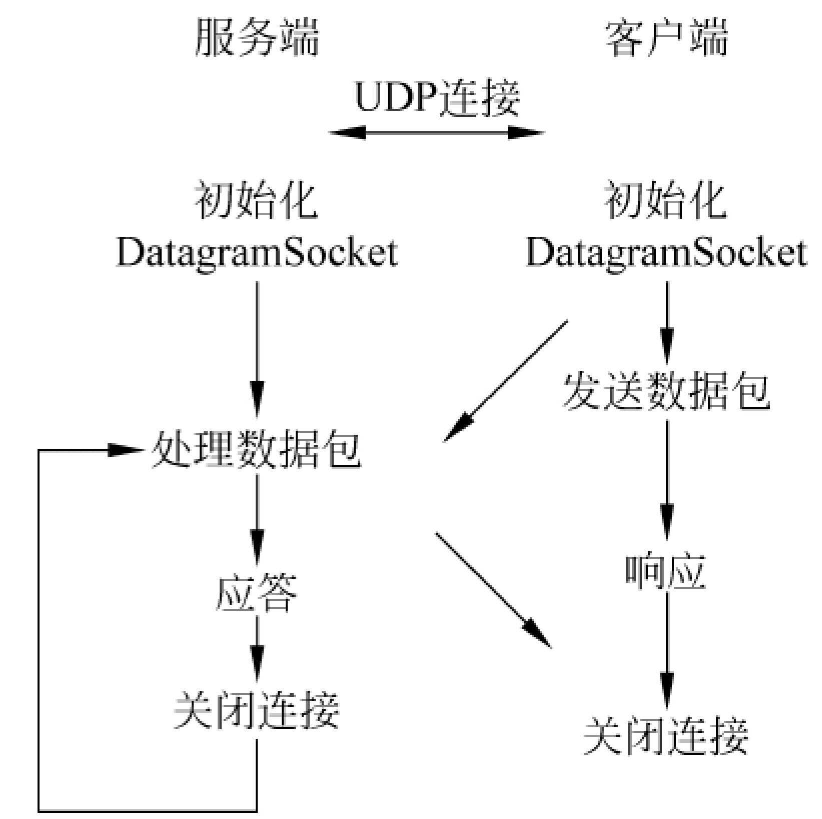


图 8-5 UDP 通信流程图

UDP 通信同样分为服务器端和客户端两个部分。

在 Android 的服务器端，主要用于开启端口、等待客户端的数据输入和应答。在服务器端实现需要以下几个步骤。

(1) 创建套接字并绑定到一个端口。在 UDP 中使用套接字 DatagramSocket 来表示数据的接收站和发送站，常用的构造函数如下：

```
DatagramSocket()
DatagramSocket(int aPort)
DatagramSocket(int aPort, InetAddress addr)
DatagramSocket(SocketAddress localAdd)
```

其中，aPort 为本地绑定的端口号，InetAddress 为指定的地址，SocketAddress 为表明绑定到特定的套接字地址。例如，创建一个监听端口号为 5000 的 UDP 套接字，实现代码为：

```
DatagramSocket = new DatagramSocket(5000)
```

(2) 接收数据。有了套接字后，即可直接使用其接收数据，使用 DatagramSocket 的实现方法为：

```
receive(DatagramPack pack)
```

其中，参数 pack 为 DatagramPacket 类型，其表示存放数据的数据包。

(3) 处理数据。无论是发送还是接收的数据都以 DatagramPack 类型表示，在处理数据前必须构造此类，但是对于接收数据包和发送数据包是有区别的。常用的接收数据的构造函数为：

```
DatagramPack(byte[] data, int length)
```

其中，参数 data 为接收的数据，length 为数据的长度。例如，创建一个可以存放 1024 个字

节数据的接收数据包，代码为：

```
byte buf[ ] = new byte[1024];
DatagramPack dp = new DatagramPack(buf,1024);
```

常用的发送数据的构造函数如下：

```
DatagramPack(byte[ ] data,int length,InetAddress host,int port)
DatagramPack(byte[ ] data,int length,SocketAddress sockAddr)
```

其中，参数 data 为发送的数据，length 为数据的长度，InetAddress 为发送到的目标地址，port 为发送到的目标端口，SocketAddress 为发送到的指定套接字地址。在 DatagramPack 类中可以获取该包发送地址的 IP 地址、端口、套接字地址以及数据内容，分别使用以下方法：

```
InetAddress getAddress()
Int getPort()
SocketAddress getSocketAddress()
byte[ ] getData()
```

对于 Android 的客户端而言，在 UDP 中，发送数据和接收数据的流程类似，都是通过套接字 DatagramSocket 发送或接收数据 DatagramPack。实现步骤如下：

(1) 创建套接字 DatagramSocket，和接收端完全一致。

(2) 发送数据 DatagramPack。

在发送端，数据包为发送数据包，必须指定数据包发送到的目标地址和端口，使用 DatagramPack 的发送构造数据包。例如，数据包目标地址为 IP 值，端口为 5000 的数据，实现代码为：

```
DatagramPack dp = new DatagramPack(buf,buf.length,InetAddress.getByName(ip),5000);
```

数据构造好后，使用 DatagramSocket 发送数据的方法为：

```
send(DatagramPack pack)
```

8.3.3 UDP 通信实例

下面通过一个实例演示 UDP 通信的使用。

【例 8-3】 使用 UDP 通信实现消息的发送。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_3UDP_send。

(2) 打开 res\layout 目录下的 main.xml 文件，在该文件中布局一个 ScrollView 控件，在该控件中声明一个 TextView 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
```

```
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "@drawable/kp" >
    <ScrollView android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
        <TextView android:id = "@ + id/mainTextView"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"/>
    </ScrollView>
</RelativeLayout>
```

(3) 打开 src\fs.li8_3udp_send 包下的 MainActivity.java 文件，用于实现接收 UDP 消息，并在一个 TextView 中显示。代码为：

```
package fs.li8_3udp_send;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.widget.TextView;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.SocketOptions;
import java.net.UnknownHostException;
public class MainActivity extends Activity {
    /* 第一次调用 Activity */
    TextView mainTextView;
    Thread mReceiveThread;
    DatagramSocket server;
    int mMessageCountInt = 0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mainTextView = (TextView) findViewById(R.id.mainTextView);
        // 定义 UDP 监听
        try {
            server = new DatagramSocket(54321);
        } catch (SocketException e) {
            // TODO 自动生成的方法存根
            e.printStackTrace();
        }
        Log.e("myLog","activity run");
        mReceiveThread = new Thread(updateThread);
        mReceiveThread.start();
        mainTextView.append("开始接收数据: \n");
    }
    @Override
    protected void onResume() {
```

```
        super.onResume();
        if(!mReceiveThread.isAlive()){
            if(server.isClosed()){
                Log.e("myLog","Resume thread");
                // 定义 UDP 监听
                try {
                    server = new DatagramSocket(54321);
                } catch (SocketException e) {
                    // TODO 自动生成的方法存根
                    e.printStackTrace();
                }
            }
            mReceiveThread = new Thread(updateThread);
            mReceiveThread.start();
        }
    }
    @Override
    protected void onPause() {
        super.onPause();
        server.close();
    }
    // 使用匿名内部类重写 Handler 中的 handlerMessage()方法
    final Handler updateBarHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            Log.e("myLog","handleMessage");
            //每次只保存 30 个
            if(mMessageCountInt++>= 30){
               mainTextView.setText("");
                mMessageCountInt = 0;
            }
            mainTextView.append((msg.getData()).getString("data"));
        }
    };
    // 线程类,该类使用匿名内部类的方式进行声明
    Runnable updateThread = new Runnable() {
        public void run() {
            // TODO  自动生成的方法存根
             Log.e("myLog","执行 run");
            // 得到一个消息对象,Message 类是 Android 系统提供的
            Message msg = new Message();
            Bundle b = new Bundle();
                try {
                    // 定义缓冲区
                    byte[] buffer = new byte[1024];
                    // 定义接收数据包
                    DatagramPacket packet = new DatagramPacket(buffer,
                            buffer.length);
                    while (true) {
                        msg = updateBarHandler.obtainMessage();
                        // 接收数据
                        server.receive(packet);
                        // 判断是否收到数据,然后输出字符串
                        if (packet.getLength() > 0) {
```

```
                            String str = new String(buffer, 0, packet
                                    .getLength());
                            b.putString("data", str + "\n");
                            msg.setData(b);
                            // 将 Message 对象加入到消息队列当中
                            updateBarHandler.sendMessage(msg);
                            Log.e("myLog", "sendMessage");
                        }
                    }
                } catch (SocketException e) {
                    Log.e("DEBUG_TAG", e.toString());
                } catch (IOException e) {
                    Log.e("DEBUG_TAG", e.toString());
                }
            }
        };
    }
```

(4) 在 src\fs.li8_3udp_send 包下创建一个名为 udpSender.java 文件，用于实现向指定 IP 发送消息"Hello+i(动态自增值)"。代码为：

```
package fs.li8_3udp_send;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
public class udpSender {
    public static void main(String[] args) {
        try {
            //定义要发送的字符串并转换为 byte 数组
            byte[] buffer = "Hello".getBytes();
            int messageCount = 0;
            //目标 ip 地址
            InetAddress wiFiDestIP = InetAddress.getByName("192.168.43.1");
            //定义 UDP 数据包，需要指定目标地址及端口号
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length, wiFiDestIP,
54321);
            //发送数据
            DatagramSocket sendSocket = new DatagramSocket();
            StringBuffer dataString = new StringBuffer();
            while(true){
                sendSocket.send(packet);
                System.out.println("Send Data:" + new String(packet.getData()));
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // TODO 自动生成的 catch 块
                    e.printStackTrace();
                }
                messageCount++;
```

```
                    buffer = ("Hello" + messageCount).getBytes();
                    packet.setData(buffer,0,buffer.length);
                }
            } catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (SocketException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

(5) 打开 AndroidManifest.xml 文件，将代码修改为：

```
…
   <activity
            android:name = "fs.li8_3udp_send.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
< uses - permission android:name = "android.permission.INTERNET"/>
</manifest >
```

运行程序，输出效果如图 8-6 所示。

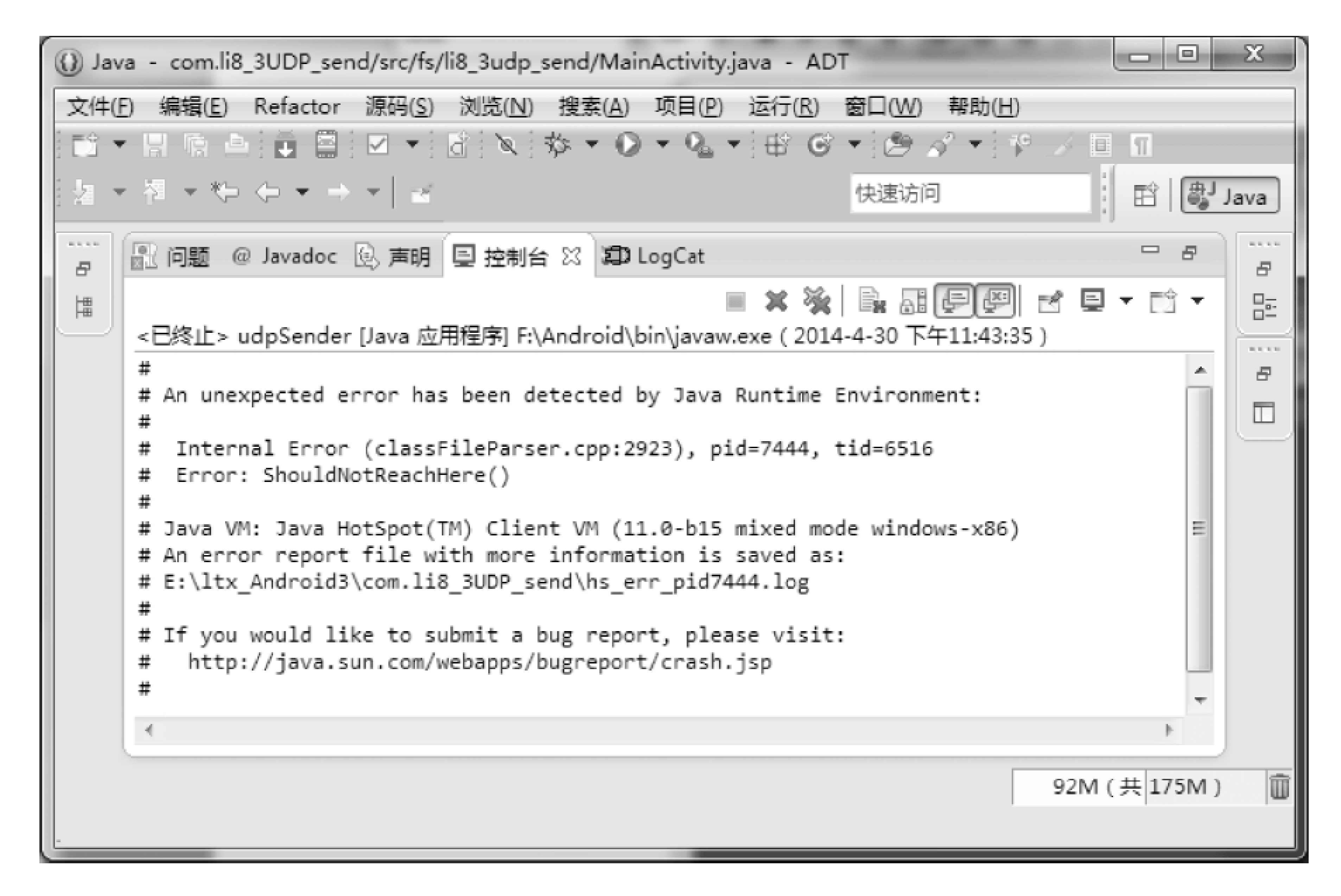


图 8-6 UDP 通信输出信息

8.4 HTTP 通信

最常用的 HTTP 请求分为 GET 和 POST 两类。GET 请求可以获得静态页面，也可以把参数放在 URL 字符串后面传递给服务器；POST 和 GET 的不同之处在于，POST 的参数不是放在 URL 字符串里面，而是放在 HTTP 请求的正文内。在 Android 中可以使用 HttpURLConnection 发送这两种请求。接下来，分别通过这两种方式获取手机号码的归属地等信息。

8.4.1 GET 请求

在 Android 中使用 GET 方式连接发送 HTTP 请求，使用的是 Java 的标准类，大家应该比较熟悉，其操作也比较简单，通过以下几步即可实现。

1. 构造 URL

在访问网络时都是通过 URL 来标定目标位置的，构造一个 URL 实例，使用以下方法：

```
URL(String spec)
```

其中，参数 spec 为 URL 地址的字符串。值得注意的是，由于使用 GET 方式发送请求，请求的参数放在 URL 字符串后面传递给服务器，即直接访问的是查询结果的网页。例如，在百度页面中，搜索“Android”，搜索结果显示的网址即为 http://www. baidu. com/s? wd=Android。因此，在 GET 中访问 URL 地址就应该是 http://www. baidu. com/s? wd=Android。

2. 设置连接

在 URL 连接中，使用 URLConnection 类来定义一个连接。当知道了访问的网络地址后，需要获取一个 URL 连接实例，使用 URL 类的方法如下：

```
URLConnection openConnectin()
```

该方法返回不同的 URLConnection 子类的对象。在本实例中，URL 是一个 HTTP 地址，因此实际返回的是 HttpURLConnection。此时，可对连接进行设置。在 GET 方式中，一般只设置连接超时时间。例如：

```
Void setReadTimeout(int timeout)
```

其中，参数为超时时间，以毫秒计算。对于是否已经连接到目标地址，通过远程 HTTP 服务器返回的响应代码进行判断。获取响应代码的方法如下：

```
int getResponseCode()
```

其中，返回值为响应编号。经常使用的值 HTTP_OK 表示已经连接；HTTP_NOT_FOUND 表示没有找到网址，等等。

3. 获取返回数据

当请求发送连接成功后，HTTP 服务器会将应答数据返回输入流中，使用 InputStreamReader 来读取返回的数据。获取返回的输入流，使用 HttpURLConnection 类

的方法如下：

```
InputStream getInputStream()
```

其中，返回值为一个输入流。由于网页采用的是 UTF-8 的编码方式，所以在读取返回的输入流时使用以下方法：

```
InputStreamReader(InputStream in,String enc)
```

其中，参数 enc 为编码方式。这里使用 UTF-8 编码。

使用这种方式查询获取手机号码的基本信息，具体实现步骤如下：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 HTTP_GET。

（2）打开 res\layout 目录下的 main.xml 文件，在该文件中实现两个 LinearLayout 布局，并声明一个 EditText 控件、一个 ScrollView 控件、一个 TextView 控件和三个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical" android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp" >
    <LinearLayout android:layout_width = "match_parent"
        android:layout_height = "wrap_content">
        <EditText android:id = "@ + id/editText"
         android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
             android:layout_weight = "1"
            android:hint = "输入所需查询的电话号码"
            android:inputType = "text">
            <requestFocus></requestFocus>
        </EditText>
        <Button android:layout_width = "match_parent"
         android:text = "Search"
            android:layout_height = "wrap_content"
            android:id = "@ + id/goQuery"
            android:layout_weight = "3"/>
    </LinearLayout>
    <Button android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/Button01"
        android:text = "使用 GET 方式获取信息"/>
    <Button android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
         android:id = "@ + id/Button02"
        android:text = "使用 POST 方式获取信息"/>
    <ScrollView android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:id = "@ + id/ScrollView1"
        android:scrollbars = "vertical">
        <TextView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:id = "@ + id/TextView01"/>
```

```
    </ScrollView>
</LinearLayout>
```

(3) 打开 src\fs.http_get 包下的 MainActivity.java 文件，在文件中实现用 GET 获取信息。代码为：

```
public class MainActivity extends Activity {
    Button btn_get, btn_post, btn_search;
    EditText edt_input;
    TextView tv_result;
//查询手机号码信息的基本网址,用于和需要查询的手机号码的拼接
    final static String phoneUrl = "http://api.showji.com/Locating/default.aspx";
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn_get = (Button) findViewById(R.id.Button01);
        btn_post = (Button) findViewById(R.id.Button02);
        btn_search = (Button) findViewById(R.id.goQuery);
        edt_input = (EditText) findViewById(R.id.editText);
        tv_result = (TextView) findViewById(R.id.TextView01);
        btn_get.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO 自动生成的方法存根
                Get_url();
            }
        });
        btn_post.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO 自动生成的方法存根
                Post_url();
            }
        });
    }
// 使用 GET 连接查询
    private void Get_url() {
        try {
            // URL
            String phonenum = edt_input.getText().toString();
            phonenum = phonenum.replace(" ", "%20");
            URL geturl = new URL( phoneUrl + "?m=" + phonenum + "&output=xml");
      // 根据拼凑的 URL 打开连接,URL.openConnection 函数会根据 URL 的类型
      // 返回不同的 URLConnection 子类的对象,这里的 URL 是一个 HTTP,因此实际返回的
      // 是 HttpURLConnection
            HttpURLConnection httpconn = (HttpURLConnection) geturl
                    .openConnection();
            httpconn.setReadTimeout(10000);    //设置超时时间
            if (httpconn.getResponseCode() == HttpURLConnection.HTTP_OK) {
                Toast.makeText(getApplicationContext(),
                        "GET 连接 手机在线 API 成功!", 1000).show();
                // InputStreamReader 得到数据流
                InputStreamReader isr = new InputStreamReader(httpconn
                        .getInputStream(), "utf-8");
```

```
                int i;
                String content = "";
                // read
                while ((i = isr.read()) != -1) {
                    content = content + (char) i;
                }
                isr.close();

                tv_result.setText(content);
            }
            // 关闭连接
            httpconn.disconnect();
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), "GET 连接 手机在线 API 失败",
                    1000).show();
            e.printStackTrace();
        }
    }
```

(4) 在实现网络请求前,必须在 AndroidManifest.xml 文件中申请权限。代码为:

```
            …
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name = "android.permission.INTERNET"/>
</manifest>
```

运行程序,效果如图 8-7 所示。



图 8-7　HTTP 通信

8.4.2 POST 请求

POST 方式相对 GET 方式而言要复杂一些，因为该方式需要将请求的参数放在 HTTP 请求的正文内，所以需要构造请求的报文。POST 方式的步骤和 GET 方式相同，只是需要对连接进行复杂的设置。

1. 构造 URL

其方法和 GET 的方法一样，不过 URL 地址是不带参数的。仍然以在百度页面中搜索"Android"为例，此时的 URL 地址为百度的网址 http：www.baidu.com。本实例中访问的 URL 为：

```
URL geturl = new URL("http://api.showji.com/Locating/default.aspx");
```

2. 设置连接

在 GET 方式中，获取连接类 URLConnection 后，使用了 URLConnection 的默认设置，不需要再对设置进行修改，而在 POST 方式中，需要更改设置为：

```
setDoOutput(true)
setDoInput(true)
```

这两个方法分别用来设置是否向 URLConnection 连接输出和输入。由于在 POST 请求中查询的参数在 HTTP 的正文内，所以需要进行输入和输出。因此，将这两个方法设置为 true。

```
setRequestMethod("POST")
```

该方法用来设置请求的方式，默认为 GET 方式，需要将其设置为 POST 方式。

```
setUseCaches(false)
```

该方法用来设置是否使用缓存，在 POST 请求中不能使用缓存，将其设置为 false。

```
setRequestProperty("Content - Type","application/x-www - form - urlencoded")
```

该方法用来设置请求正文的类型。由于在正文内容中将使用 URLEncoder.encode 进行编码，所以设置如上，表示正文是 urlencoded 编码过的 from 参数。

完成这些设置后，即可连接到远程 URL，使用的方法为：

```
connect()
```

3. 写入请求正文

在 POST 方式中，需要将请求的内容写在请求正文中发送到远程服务器。首先需要获取连接的输出流，使用方法为：

```
OutputStream getOutputStream()
```

获取了输出流后，需要将参数写入该输出流中，写入的内容和 GET 方式中 URL 的"?"后的参数字符串一致。值得注意的是，对于从输入框中输入的查询电话号码必须进行 URL 编码。例如，在本实例中，写入的内容为：

```
String conent = "m = " + URLEncoder.encode(phonenum,"utf - 8") + "&output = xml";
```

4. 读取返回数据、关闭连接

完成数据的请求后,读取返回数据和关闭连接的方法和 GET 请求方式是一样的。使用 POST 方式来查询获取手机号码的基本信息。

打开 src\fs.http_get 包下的 MainActivity.java 文件,添加以下代码:

```
    // POST 方式
//查询手机号码信息的 URL 地址
final static String phoneUrl = "http://api.showji.com/Locating/default.aspx";
    private void Post_url() {
        String phonenum = edt_input.getText().toString();
  //构造 URL,直接使用查询信息的网址
        try {
            // 构造一个 URL 对象
            URL url = new URL(phoneUrl);
            // 使用 HttpURLConnection 打开连接
            HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
            // 因为这个是 POST 请求,需要设置为 true
            urlConn.setDoOutput(true);
            urlConn.setDoInput(true);
            //设置超时时间
            urlConn.setReadTimeout(10000);
            // 设置 POST 方式
            urlConn.setRequestMethod("POST");
            // POST 请求不使用缓存
            urlConn.setUseCaches(false);
            urlConn.setInstanceFollowRedirects(true);
            // 配置本次连接的 Content - Type,配置为 application/x - www - form - urlencoded
            urlConn.setRequestProperty("Content - Type",
                    "application/x - www - form - urlencoded");
            // 连接,从 postUrl.openConnection()至此的配置必须在 connect 之前完成
            // 需要注意的是,connection.getOutputStream 会隐含地进行 connect
            urlConn.connect();
            // DataOutputStream 流
            DataOutputStream out = new DataOutputStream(urlConn.getOutputStream());
            // 要上传的参数,GET 方式"?"之后的内容.
             m = " + phonenum + " &output = xml" String content = "m = " + URLEncoder.encode
(phonenum, "utf - 8") + "&output = xml";
            // 将要上传的内容写入流中
            out.writeBytes(content);
            // 刷新、关闭
            out.flush();
            out.close();
            InputStreamReader isr = new InputStreamReader(urlConn.getInputStream());
            int i;
            String content_post = "";
            // read
            while ((i = isr.read()) != - 1) {
                content_post = content_post + (char) i;
            }
            isr.close();
```

```
            // 设置 TextView
            tv_result.setText(content_post);
            // 关闭 HTTP 连接
            urlConn.disconnect();
            Toast.makeText(getApplicationContext(), "POST 连接手机在线 API 成功",
                    1000).show();
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), "POST 连接手机在线 API 失败",
                    1000).show();
            e.printStackTrace();
        }
    }
}
```

8.5 WebView 浏览器

在 Android 中可以很容易地实现一个定制的浏览器，因为 Android 提供了 WebView 控件专门用来浏览网页，使用非常方便。本节将使用 WebView 控件来实现定制的浏览器，使浏览器具有网页拍照功能。它使用了 WebKit 渲染引擎加载显示网页，实现 WebView 有以下两种不同的方法。

第一种方法的步骤如下。

(1) 在 Activity 中实例化 WebView 控件：

```
WebView webView = new WebView(this);
```

(2) 调用 WebView 的 loadUrl()方法，设置 WevView 要显示的网页。

互联网用：webView.loadUrl("http://www.google.com")；

本地文件用：webView.loadUrl("file:///android_asset/XX.html")；

本地文件存放在 assets 文件中。

(3) 调用 Activity 的 setContentView()方法显示网页视图。

(4) 用 WebView 单击链接看了很多页以后，为了让 WebView 支持回退功能，需要覆盖 Activity 类的 onKeyDown()方法，如果不做任何处理，单击系统回退键，整个浏览器会调用 finish()结束自身，而不是回退到上一页面。

(5) 需要在 AndroidManifest.xml 文件中添加权限，否则会出现 Web page not available 错误。

```
<uses-permission android:name="android.permission.INTERNET" />
```

其具体实现的实例如下。

(1) 在 Eclipse 中创建一个 Andriod 应用项目，命名为 WebView_1。

(2) 打开 src\fs.webview_1 包下的 MainActivity.java 文件，代码为：

```
package fs.webview_1;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.webkit.WebView;
```

```
public class MainActivity extends Activity {
    private WebView webview;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //实例化 WebView 对象
        webview = new WebView(this);
        //设置 WebView 属性,能够执行 JavaScript 脚本
        webview.getSettings().setJavaScriptEnabled(true);
        //加载需要显示的网页
        webview.loadUrl("http://www.51cto.com/");
        //设置 Web 视图
        setContentView(webview);
        }
    @Override
    //设置回退
    //覆盖 Activity 类的 onKeyDown(int keyCoder,KeyEvent event)方法
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if ((keyCode == KeyEvent.KEYCODE_BACK) && webview.canGoBack()) {
            webview.goBack();
            //goBack()表示返回 WebView 的上一页面
            return true;
            }
        return false;
        }
}
```

(3) 打开 AndroidManifest.xml 文件,添加对应的权限。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.webview_1"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.webview_1.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
< uses - permission android:name = "android.permission.INTERNET"/>
</manifest >
```

运行程序,效果如图 8-8 所示。



图 8-8　用第一种方法浏览网页

第二种方法的步骤如下。

(1) 在布局文件中声明 WebView。

(2) 在 Activity 中实例化 WebView。

(3) 调用 WebView 的 loadUrl()方法，设置 WebView 要显示的网页。

(4) 为了让 WebView 能够响应超链接功能，调用 setWebViewClient()方法设置 WebView 视图。

(5) 用 WebView 单击链接看了很多页以后，为了让 WebView 支持回退功能，需要覆盖 Activity 类的 onKeyDown()方法，如果不做任何处理，单击系统回退键，整个浏览器会调用 finish()结束自身，而不是回退到上一页面。

(6) 需要在 AndroidManifest.xml 文件中添加权限，否则会出现 Web page not available 错误。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

其具体实现的实例如下。

(1) 在 Eclipse 中创建一个 Andriod 应用项目，命名为 WebView_2。

(2) 打开 src\fs.webview_2 包下的 MainActivity.java 文件，代码为：

```
package fs.webview_2;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.webkit.WebView;
```

```java
import android.webkit.WebViewClient;
public class MainActivity extends Activity {
    private WebView webview;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        webview = (WebView) findViewById(R.id.webview);
        //设置 WebView 属性,能够执行 JavaScript 脚本
        webview.getSettings().setJavaScriptEnabled(true);
        //加载需要显示的网页
        webview.loadUrl("http://www.51cto.com/");
        //设置 Web 视图
        webview.setWebViewClient(new HelloWebViewClient ());
        }
    @Override
    //设置回退
    //覆盖 Activity 类的 onKeyDown(int keyCoder,KeyEvent event)方法
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if ((keyCode == KeyEvent.KEYCODE_BACK) && webview.canGoBack()) {
            webview.goBack();
            //goBack()表示返回 WebView 的上一页面
            return true;
            }
        return false;
        }
    //Web 视图
    private class HelloWebViewClient extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            view.loadUrl(url);
            return true;
            }
        }
}
```

(3) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 WebView 控件。代码为:

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
   <WebView
       android:id="@+id/webview"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent" />
</RelativeLayout>
```

(4) 打开 AndriodManifest.xml 文件,添加相应的权限。代码为:

```
    …
            < action android:name = "android.intent.action.MAIN" />
                  < category android:name = "android.intent.category.LAUNCHER" />
              </intent - filter >
          </activity >
      </application >
< uses - permission android:name = "android.permission.INTERNET"/>
</manifest >
```

运行程序,效果如图 8-9 所示。



图 8-9 用第二种方法浏览网页

8.6 手机通信综合实例

下面利用 Android 实现一个在线翻译以及理解实例。

【例 8-4】 在线翻译以及理解实例。当大家遇到不认识的单词或不理解的词语时,一般会借助于网络来进行查询。具体的实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li8_4query。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明两个 TextView 控件、一个 EditText 控件、两个 RadioGroup 控件、一个 WebView 控件和一个 Button 控件。代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "@drawable/kp">
<TextView
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_centerHorizontal = "true"
        android:text = "在线查询" />
    <EditText
        android:id = "@ + id/tinput"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/textView1"
        android:ems = "10"
        android:hint = "输入要查询的词" />
    <RadioGroup
        android:id = "@ + id/myRadioGroup"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentRight = "true"
        android:layout_below = "@ + id/tinput"
        android:orientation = "horizontal" >
        <RadioButton
            android:id = "@ + id/myRadioButton1"
            android:layout_height = "wrap_content"
            android:text = "翻译" />
        <RadioButton
            android:id = "@ + id/myRadioButton2"
            android:layout_height = "wrap_content"
            android:text = "百科" />
    </RadioGroup>
    <Button
        android:id = "@ + id/submit"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentRight = "true"
        android:layout_below = "@ + id/myRadioGroup"
        android:text = "查询" />
    <TextView
        android:id = "@ + id/tips"
        android:layout_width = "fill_parent"
```

```
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/submit"
        android:text = "查询结果: "
        android:textSize = "14sp"
        android:visibility = "invisible"
        android:typeface = "sans" />
    < WebView
        android:id = "@ + id/toutput"
        android:layout_width = "fill_parent"
        android:layout_height = "270px"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/tips"
        android:visibility = "invisible" />
</RelativeLayout >
```

(3) 打开 src\fs. li8_4query 包下的 MainActivity. java 文件,在文件中实现单词的查询与理解。代码为:

```
package fs. li8_4query;
import android. os. Bundle;
import android. os. Handler;
import android. app. Activity;
import android. view. Menu;
import android. view. View;
import android. view. View. OnClickListener;
import android. webkit. WebSettings;
import android. webkit. WebView;
import android. webkit. WebViewClient;
import android. widget. Button;
import android. widget. EditText;
import android. widget. RadioButton;
import android. widget. RadioGroup;
import android. widget. TextView;
import android. widget. Toast;
public class MainActivity extends Activity {
    private TextView tips;
    private EditText editText;
    private WebView webView;
    private Button submit;
    RadioButton rb1, rb2;
    RadioGroup rGroup;
    private Handler tHandler = new Handler();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);
        webView = (WebView) findViewById(R. id. toutput);
        submit = (Button) findViewById(R. id. submit);
        editText = (EditText) findViewById(R. id. tinput);
```

```
            tips = (TextView) findViewById(R.id.tips);
            rb1 = (RadioButton) findViewById(R.id.myRadioButton1);
            rb2 = (RadioButton) findViewById(R.id.myRadioButton2);
            rGroup = (RadioGroup) findViewById(R.id.myRadioGroup);
            rGroup.check(R.id.myRadioButton1);
            WebSettings webSettings = webView.getSettings();
            webSettings.setSaveFormData(false);
            webSettings.setSavePassword(false);
            webSettings.setSupportZoom(false);
            webView.setWebViewClient(new WebViewClient() {
                @Override
                public boolean shouldOverrideUrlLoading(
                        WebView view, String url) {
                    // 使用自己的 WebView 加载
                    view.loadUrl(url);
                    return true;
                }
            });
            submit.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (editText.getText().toString().equals("")) {
                        Toast.makeText(MainActivity.this, "请输入查询的词",
                                Toast.LENGTH_LONG);
                        return;
                    }
                    tips.setVisibility(TextView.VISIBLE);
                    webView.setVisibility(WebView.VISIBLE);
                    tHandler.post(new Runnable() {
                        public void run() {
                    if (rGroup.getCheckedRadioButtonId() == R.id.myRadioButton1) {
                                webView.loadUrl("http://3g.dict.cn/s.php?q="
                                        + editText.getText().toString());
                            } else {
                                webView.loadUrl("http://www.baike.com/wiki/"
                                        + editText.getText().toString());
                            }
                        }
                    });
                }
            });
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // 增加了项目操作栏
            getMenuInflater().inflate(R.menu.main, menu);
            return true;
        }
    }
```

运行程序,效果如图 8-10 所示。



图 8-10　查询界面

8.7　手机服务

作为一个移动手机设备，当然需要具备交互通信的功能。手机中的交互通信方式有多种，例如常见的通话、短信、邮件及 WiFi 等。

8.7.1　电话拨打功能

拨打电话是手机最平常不过的事了，拨打电话在 Android 中是怎样实现的呢？本节将通过一个实例来介绍。

"打电话"功能是每个手机所具有的最基本的功能，在本实例中是以 EditText 为输入电话号码的编辑框，当单击"拨打"按钮后，实现拨打电话的功能。当然，为了避免用户输入的电话号码格式错误，在单击"拨打"按钮时，判断用户所输入的是否为正确的电话号码，如果号码不正确，则提示用户电话号码的输入格式错误，并将 EditText 中的文本设置为空字符。

【例 8-5】 实现电话的拨打功能。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_5phone。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 TextView 控件、一个 EditText 控件和一个 Button 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
```

```
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"
        android:paddingBottom = "@dimen/activity_vertical_margin"
        android:paddingLeft = "@dimen/activity_horizontal_margin"
        android:paddingRight = "@dimen/activity_horizontal_margin"
        android:paddingTop = "@dimen/activity_vertical_margin"
        tools:context = ".MainActivity"
        android:background = "@drawable/kp">
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "请输入你的号码"/>
        <EditText
        android:id = "@ + id/phonenumber"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:phoneNumber = "true" />
        <Button
            android:id = "@ + id/btn_call"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentLeft = "true"
            android:layout_below = "@ + id/phonenumber"
            android:layout_marginTop = "120dp"
            android:text = "拨打" />
    </RelativeLayout>
```

(3) 打开 src\fs.li8_5phone 包下的 MainActivity.java 文件,在文件中实现电话号码的拨打功能。代码为:

```
package fs.li8_5phone;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity {
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn_call = (Button) findViewById(R.id.btn_call);
        btn_call.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // TODO 自动生成的方法存根
                EditText et_phonenumber = (EditText)findViewById(R.id.phonenumber);
                String number = et_phonenumber.getText().toString();
                //用 Intent 启动拨打电话
                Intent intent = new Intent(Intent.ACTION_CALL,Uri.parse("tel:" + number));
                startActivity(intent);
```

```
            }
        });
    }
}
```

(4) 打开 AndroidManifest.xml 文件,设置拨打电话权限。代码为:

```
…
   <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <!-- 添加可以向外拨打电话的权限 -->
    <uses-permission android:name="android.permission.CALL_PHONE"/>
…
```

运行程序,效果如图 8-11 所示。



图 8-11　拨打号码界面

8.7.2　自制电话拨号功能

Android 系统自带了一套拨号系统,在 Android 中是允许对其进行更改的。本例将要为读者介绍怎样替换系统的拨号系统,即使用自定义的个性拨号系统。

【例 8-6】 对于没有键盘的手机,每次拨打电话总会使用系统自带的拨号按钮,如果觉得手机的拨号按钮不好看,用户也可自己制作一个个性的拨号系统,为每一个按钮添加监听器,当单击按钮时,在 EditText 中显示所单击的数字,最后按拨号键完成电话的拨号功能。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li8_6CustomPhone。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中实现 6 个线性布局,声明 1 个 EditText 控件及 10 个 Button 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp">
  <LinearLayout
    android:id = "@ + id/LinearLayout6"
    android:orientation = "horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content">
   <EditText
      android:text = "@string/default_number"
      android:id = "@ + id/EditText1"
      android:layout_width = "260dip"
      android:textSize = "24dip"
      android:editable = "false"
      android:enabled = "false"
      android:singleLine = "true"
      android:background = "#FFFFFF"
      android:textColor = "#000000"
      android:layout_marginRight = "6dip"
      android:layout_marginLeft = "10dip"
      android:layout_height = "wrap_content"/>
   <Button
      android:text = " "
      android:id = "@ + id/Button_del"
      android:textSize = "24dip"
      android:layout_width = "wrap_content"
      android:layout_height = "wrap_content"
      android:background = "@drawable/deldown"/>
  </LinearLayout>
  <LinearLayout
    android:id = "@ + id/LinearLayout1"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content">
   <LinearLayout
      android:id = "@ + id/LinearLayout2"
      android:orientation = "horizontal"
      android:gravity = "center_horizontal"
      android:layout_width = "fill_parent"
      android:layout_height = "wrap_content">
        <Button
          android:text = "1"
          android:id = "@ + id/Button1"
          android:textSize = "54dip"
          android:textStyle = "bold"
          android:typeface = "serif"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
```

```
            android:background = "@drawable/c1"/>
        <Button
            android:text = "2"
            android:id = "@ + id/Button2"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "20dip"
            android:layout_marginRight = "20dip"
            android:background = "@drawable/c1"/>
        <Button
            android:text = "3"
            android:id = "@ + id/Button3"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/c1"/>
</LinearLayout>
<LinearLayout
    android:id = "@ + id/LinearLayout3"
    android:orientation = "horizontal"
    android:gravity = "center_horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:layout_marginTop = "20dip">
        <Button
            android:text = "4"
            android:id = "@ + id/Button4"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/c1"/>
        <Button
            android:text = "5"
            android:id = "@ + id/Button5"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "20dip"
            android:layout_marginRight = "20dip"
            android:background = "@drawable/c1"/>
        <Button
            android:text = "6"
            android:id = "@ + id/Button6"
            android:textSize = "54dip"
            android:textStyle = "bold"
```

```
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/c1"/>
    </LinearLayout>
    <LinearLayout
        android:id = "@ + id/LinearLayout4"
        android:orientation = "horizontal"
        android:gravity = "center_horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_marginTop = "20dip">
          <Button
            android:text = "7"
            android:id = "@ + id/Button7"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/c1"/>
          <Button
            android:text = "8"
            android:id = "@ + id/Button8"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "20dip"
            android:layout_marginRight = "20dip"
            android:background = "@drawable/c1"/>
          <Button
            android:text = "9"
            android:id = "@ + id/Button9"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/c1"/>
    </LinearLayout>
    <LinearLayout
        android:id = "@ + id/LinearLayout5"
        android:orientation = "horizontal"
        android:gravity = "center_horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_marginTop = "20dip">
          <Button
            android:text = " "
            android:id = "@ + id/Button_dial"
            android:textSize = "54dip"
            android:textStyle = "bold"
```

```
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/dial"/>
        < Button
            android:text = "0"
            android:id = "@ + id/Button0"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "20dip"
            android:layout_marginRight = "20dip"
            android:background = "@drawable/ic_launcher"/>
        < Button
            android:text = " "
            android:id = "@ + id/Button_cancel"
            android:textSize = "54dip"
            android:textStyle = "bold"
            android:typeface = "serif"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/dialcancel"/>
    </LinearLayout >
  </LinearLayout >
</LinearLayout >
```

(3) 打开 src\fs. li8_6customphone 包下的 MainActivity. java 文件，实现自定义拨号功能。代码为：

```
package fs. li8_6customphone;
import android. app. Activity;
import android. content. Intent;
import android. net. Uri;
import android. os. Bundle;
import android. view. View;
import android. widget. Button;
import android. widget. EditText;
public class MainActivity extends Activity {
    //数字按钮的 id 数组
    int[ ] numButtonIds =
    {
        R. id. Button0, R. id. Button1, R. id. Button2,
        R. id. Button3, R. id. Button4, R. id. Button5,
        R. id. Button6, R. id. Button7, R. id. Button8,
        R. id. Button9
    };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);
        //为删除按钮添加监听器
        Button bDel = (Button)this. findViewById(R. id. Button_del);
```

```
bDel.setOnClickListener(
   //OnClickListener 为 View 的内部接口,其实现者负责监听鼠标单击事件
        new View.OnClickListener()
        {
         public void onClick(View v)
         {
             EditText et = (EditText)findViewById(R.id.EditText1);
             String num = et.getText().toString();
             num = (num.length()> 1)?num.substring(0,num.length() - 1):"";
             et.setText(num);
         }
});
//为拨号按钮添加监听器
Button bDial = (Button)this.findViewById(R.id.Button_dial);
bDial.setOnClickListener(
 //OnClickListener 为 View 的内部接口,其实现者负责监听鼠标单击事件
        new View.OnClickListener()
        {
         public void onClick(View v)
         {
             //获取输入的电话号码
             EditText et = (EditText)findViewById(R.id.EditText1);
             String num = et.getText().toString();
             //根据获取的电话号码创建 Intent 拨号
             Intent dial  =  new Intent();
             dial.setAction("android.intent.action.CALL");
             dial.setData(Uri.parse("tel://" + num));
             startActivity(dial);
         }
});
//为退出按钮添加监听器
Button bCancel = (Button)this.findViewById(R.id.Button_cancel);
bCancel.setOnClickListener(
   //OnClickListener 为 View 的内部接口,其实现者负责监听鼠标单击事件
        new View.OnClickListener()
        {
         public void onClick(View v)
         {
             MainActivity.this.finish();
         }
});
//为 0~9 数字按钮创建监听器
View.OnClickListener numListener = new View.OnClickListener()
{
    public void onClick(View v)
    {
     Button tempb = (Button)v;
     EditText et = (EditText)findViewById(R.id.EditText1);
     et.append(tempb.getText());
    }
};
//为所有的数字按钮添加监听器
for(int id:numButtonIds)
{
```

```
                Button tempb = (Button)this.findViewById(id);
                tempb.setOnClickListener(numListener);
            }

        }
    }
```

运行程序，效果如图 8-12 所示。



图 8-12　自定义拨号功能

8.7.3　短信功能

在 Android 中主要采用 SmsManager 的 sendTextMessage( ) 方法发送文字短信，sendTextMessage()方法有 5 个参数，第 1 个参数为对方的手机号码(不能为空)，第 2 个参数为发送方的手机号码(可以为空)，第 3 个参数为发送的短信内容(不能为空)，第 4 个参数为 PendingIntent 对象，用于判断发送短信是否成功(可以为空)，第 5 个参数也为 PendingIntent 对象，当用户接收到短信时会返回该对象(可以为空)。

下面通过一个实例来演示手机的发送短信功能。

【例 8-7】　实现用 Android 发送短信。

其操作步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_7SMS。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明两个 TextView 控件及两个 EditText 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
```

```
        xmlns:tools = "http://schemas.android.com/tools"
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"
        android:background = "@drawable/kp"
        android:paddingBottom = "@dimen/activity_vertical_margin"
        android:paddingLeft = "@dimen/activity_horizontal_margin"
        android:paddingRight = "@dimen/activity_horizontal_margin"
        android:paddingTop = "@dimen/activity_vertical_margin"
        tools:context = ".MainActivity" >
        <TextView
            android:id = "@ + id/textView1"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "@string/str_input_phone_number" />
        <EditText
            android:id = "@ + id/phone_number_editText"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <Button
            android:id = "@ + id/send_sms_button"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignLeft = "@ + id/textView2"
            android:layout_below = "@ + id/textView2"
            android:layout_marginTop = "81dp"
            android:text = "@string/str_send_sms" />
        <TextView
            android:id = "@ + id/textView2"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_alignLeft = "@ + id/phone_number_editText"
            android:layout_alignTop = "@ + id/sms_content_editText"
            android:text = "@string/str_input_sms_content" />
        <EditText
            android:id = "@ + id/sms_content_editText"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_alignLeft = "@ + id/textView2"
            android:layout_below = "@ + id/phone_number_editText"
            android:layout_marginTop = "46dp"
            android:ems = "10" >
            <requestFocus />
        </EditText>
    </RelativeLayout>
```

(3) 打开 src\fs.li8_7sms 包下的 MainActivity.java 文件,在文件中实现发送短信功能。代码为:

```
package fs.li8_7sms;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
```

```java
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
     phone_number_editText = (EditText) findViewById(R.id.phone_number_editText);
        sms_content_editText = (EditText) findViewById(R.id.sms_content_editText);
        send_sms_button = (Button) findViewById(R.id.send_sms_button);
        send_sms_button.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
            String phone_number = phone_number_editText.getText().toString().trim();
                String sms_content = sms_content_editText.getText().toString().trim();
                if(phone_number.equals("")) {
                     Toast.makeText(MainActivity.this, R.string.str_remind_input_phone_
number, Toast.LENGTH_LONG).show();
                } else {
                    SmsManager smsManager = SmsManager.getDefault();
                    if(sms_content.length() > 70) {
                        List<String> contents = smsManager.divideMessage(sms_content);
                        for(String sms : contents) {
                  smsManager.sendTextMessage(phone_number, null, sms, null, null);
                        }
                    } else {
            smsManager.sendTextMessage(phone_number, null, sms_content, null, null);
                    }
                      Toast.makeText(MainActivity.this, R.string.str_remind_sms_send_
finish, Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
    private EditText phone_number_editText;
    private EditText sms_content_editText;
    private Button send_sms_button;
}
```

(4) 打开 res\layout 目录下的 strings.xml 文件，为控件和界面添加对应的题目及标题。代码为：

```xml
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name">发短信程序</string>
    <string name = "action_settings">Settings</string>
    <string name = "hello_world">Hello world!</string>
  <string name = "str_input_phone_number">请输入手机号</string>
    <string name = "str_input_sms_content">请输入短信内容</string>
    <string name = "str_send_sms">发送</string>
    <string name = "str_remind_input_phone_number">请输入手机号</string>
```

```
    <string name = "str_remind_sms_send_finish">发送完成</string>
</resources>
```

(5) 打开 AndroidManifest.xml 文件,实现短信权限。代码为:

```
…
  <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <! -- 添加短信权限 -->
<uses - permission android:name = "android.permission.SEND_SMS"/>
…
```

运行程序,效果如图 8-13 所示。



图 8-13　手机发送短信界面

8.7.4　接收短信

用户除了可以从 Android 应用程序发送 SMS 消息外,还可以在应用程序中使用 BroadcastReceiver 对象接收传入的 SMS 消息。如果希望应用程序在收到一条特定的 SMS 消息时执行一个动作(这是很有用的,例如根据追踪用户的手机位置以防丢失或被盗)可以编写一个应用程序用来自动侦听包含一些秘密代码的 SMS 消息,一旦收到此类信息,即可给发送者发回一条包含位置坐标的 SMS 消息。

下面通过一个实例来演示 Android 手机接收短信。

【例 8-8】 实现手机接收短信功能。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li8_8Receive_SMS。

(2) 打开 AndroidManifest.xml 文件，设置接收短信权限。代码为：

```
…
   </application>
    <! -- 设置短信声明权限 -->
    < uses - permission android:name = "android.permission.RECEIVE_SMS">
    </uses - permission>
    < uses - permission android:name = "android.permission.SEND_SMS"/>
</manifest>
…
```

(3) 打开 res\layout 目录下的 main.xml 文件，用于声明两个 TextView 控件、两个 EditText 控件和一个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:orientation = "vertical"
android:background = "@drawable/kp">
    < TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "发送者号码"/>
    < EditText
        android:id = "@ + id/txtPhoneNo"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"/>
    < TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "信息内容"/>
    < EditText
        android:id = "@ + id/txtMessage"
        android:layout_width = "fill_parent"
        android:layout_height = "150px"
        android:gravity = "top"/>
    < Button
        android:id = "@ + id/btnSendSMS"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "回复短信"/>
</LinearLayout>
```

(4) 打开 src\fs.li8_8receive_sms 包下的 MainActivity.java 文件，用于实现短信的接收。代码为：

```
package fs.li8_8receive_sms;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;
public class MainActivity extends BroadcastReceiver
```

```
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //获取传入的 SMS 信息
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "";
        if (bundle != null)
        {
            //检索接收到的 SMS 消息
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i = 0; i < msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                str += "SMS from " + msgs[i].getOriginatingAddress();
                str += " :";
                str += msgs[i].getMessageBody().toString();
                str += "\n";
            }
            //显示新的 SMS 消息
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
            //MainActivity 的扩展
            Intent mainActivityIntent = new Intent(context, MainActivity.class);
            mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(mainActivityIntent);
            //发送一个广播意图来更新活动中接收到的 SMS 消息
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
        }
    }
}
```

运行程序，效果如图 8-14 所示。



图 8-14　短信接收界面

8.7.5 电子邮件

前面提到了手机的两个最基本的功能——打电话和发送短信，下面介绍手机的另外一个功能——发送 E-mail，该软件通过利用 Android 强大的网络支持能力向目标用户发送 E-mail。

在该软件的界面中需要填写的是收件人地址、发送人地址、主题以及邮件的内容，单击“发送”按钮时，软件会自动检测收件人地址和发件人地址的格式填写的是否正确，如果不正确则使用 Toast 提示用户填写错误，如果填写正确，则正常发送 E-mail。

【例 8-9】 利用 Android 发送 E-mail。

该软件通过自定义 Intent 对象，使用 Android. content. Intent. ACTION_SEND 的参数来实现通过手机发送 E-mail 服务。在发送过程中需要检测所输入的 E-mail 地址是否符合格式要求，否则不可能发送成功，然而实际上，在手机上发送或接收 E-mail 是通过 Android 内置的 Gmail 程序处理的。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_9E-mail。

(2) 打开 res\layout 目录下的 main. xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp">
    <LinearLayout
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:orientation = "horizontal">
        <TextView
          android:text = "收件人地址: "
          android:id = "@ + id/TextView1"
          android:textColor = "#222333"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"/>
        <EditText
          android:text = "sunjia_123@163.com"
          android:id = "@ + id/EditText1"
          android:textColor = "#222333"
          android:layout_width = "fill_parent"
          android:layout_height = "wrap_content"/>
    </LinearLayout>
    <LinearLayout
      android:layout_width = "fill_parent"
      android:layout_height = "wrap_content"
      android:orientation = "horizontal">
        <TextView
          android:text = "发件人地址: "
          android:id = "@ + id/TextView4"
          android:textColor = "#222333"
```

```
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"/>
            <EditText
                android:text = "thesun_123@163.com"
                android:id = "@ + id/EditText4"
                android:textColor = "#222333"
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:orientation = "horizontal">
            <TextView
                android:text = "邮件主题: "
                android:id = "@ + id/TextView2"
                android:textColor = "#222333"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"/>
            <EditText
                android:id = "@ + id/EditText2"
                android:textColor = "#222333"
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"/>
        </LinearLayout>
        <TextView
                android:text = "邮件内容: "
                android:textColor = "#222333"
                android:id = "@ + id/TextView3"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"/>
        <EditText
                android:id = "@ + id/EditText3"
                android:textColor = "#222233"
                android:layout_width = "fill_parent"
                android:layout_height = "100dip"
                android:gravity = "top|left"/>
            <Button
                android:text = "发送"
                android:textColor = "#222333"
                android:id = "@ + id/Button1"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"/>
</LinearLayout>
```

(3) 打开 res\values 目录下的 strings.xml 文件,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name">发送邮件</string>
    <string name = "action_settings"> Settings </string>
    <string name = "hello_world"> Hello world!</string>
    <string name = "start">邮件发送中……</string>
</resources>
```

(4) 打开 src\fs.li8_9e-mail 包下的 MainActivity.java 文件,实现手机发送 E-mail 的功能。代码为:

```
package fs.li8_9e_mail;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
    EditText etReceiver;                        //收件人
    EditText etSender;                          //发件人
    EditText etTheme;                           //主题
    EditText etMessage;                         //内容
    Button bSend;                               //"发送"按钮
    String strReceiver;                         //收件人信息
    String strSender;                           //发件人信息
    String strTheme;                            //主题信息
    String strMessage;                          //内容信息
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        etReceiver = (EditText)this.findViewById(R.id.EditText1);      //获取对象
        etSender = (EditText)this.findViewById(R.id.EditText4);        //获取对象
        etTheme = (EditText)this.findViewById(R.id.EditText2);         //获取对象
        etMessage = (EditText)this.findViewById(R.id.EditText3);       //获取对象
        bSend = (Button)this.findViewById(R.id.Button1);               //"发送"按钮
        bSend.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v) {
                    strReceiver = etReceiver.getText().toString().trim();   //获取收件人
                          strSender = etSender.getText().toString().trim(); //获取发件人
                        strTheme = etTheme.getText().toString().trim();     //获取主题
                        strMessage = etMessage.getText().toString().trim(); //获取内容
                        String parent = "^[a-zA-Z][\\w\\.-]*[a-zA-Z0-9]@[a-zA-
Z0-9][\\w\\.-]*[a-zA-Z0-9]\\.[a-zA-Z][a-zA-Z\\.]*[a-zA-Z]$";
                    if(!strReceiver.matches(parent))       //查看收件人地址是否符合格式
                        {
                              Toast.makeText(MainActivity.this, "收件人地址格式错误",
Toast.LENGTH_SHORT).show();
                }else if(!strSender.matches(parent))       //查看发件人地址是否符合格式
                        {
Toast.makeText(MainActivity.this, "发件人地址格式错误", Toast.LENGTH_SHORT).show();
                        }else                              //若都符合格式,则发送邮件
                        {
    Intent intent = new Intent(android.content.Intent.ACTION_SEND);    //发送邮件功能
                            intent.setType("plain/text");
                intent.putExtra(android.content.Intent.EXTRA_EMAIL, strReceiver);
```

```
                        intent.putExtra(android.content.Intent.EXTRA_CC, strSender);
                        intent.putExtra(android.content.Intent.EXTRA_SUBJECT, strTheme);
                        intent.putExtra(android.content.Intent.EXTRA_TEXT, strMessage);
            startActivity(Intent.createChooser(intent, getResources().getString(R.string.start)));
                            }
                        }

                    }
                );
            }
        }
```

运行程序,效果如图 8-15 所示。



图 8-15　发送 E-mail 界面

8.7.6　通讯录搜索

本节介绍怎样对通讯录中的联系人进行搜索,主要是对 ContentResolver 的应用。

手机中的通讯录是一个不可缺少的部分,手机用户的所有联系人均在里面,本软件实现的即为通过自制的手机通讯录软件对联系人进行简单搜索。

在 EditText 中输入所要搜索联系人的名字的首字母,在自定义的 ContentResolver 中会显示出首字母相同的所有联系人,当输入的是“*”时,则会显示出所有的联系人。当单击其中一个联系人时,会在主界面中显示该联系人的姓名和电话。如果该用户没有电话,则会显示该联系人尚未有电话号码。

【例 8-10】 实现手机通讯录的搜索功能。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_10Search。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中声明一个 TextView 控件。代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="#aaaccc">
    <AutoCompleteTextView
     android:id="@+id/AutoCompleteTextView1"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"/>
    <TextView
     android:id="@+id/TextView1"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:textColor="#222333"/>
</RelativeLayout>
```

(3) 打开 src\fs. li8_10search 包下的 MainActivity. java 文件，在文件中声明需要查询的通讯录字段名，并显示搜索结果。代码为：

```
package fs.li8_10search;
import android.app.Activity;
import android.content.ContentResolver;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.Contacts;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AutoCompleteTextView;
import android.widget.TextView;
public class MainActivity extends Activity {
    private AutoCompleteTextView Aclt;                    //声明引用
    private TextView tv;
    private Cursor cursor;
    private ContactsAdapter Ca;
    static final String[] PEOPLE_PROJECTION =
    {
        Contacts.People._ID,
        Contacts.People.PRIMARY_PHONE_ID,
        Contacts.People.TYPE,
        Contacts.People.NUMBER,
        Contacts.People.LABEL,
        Contacts.People.NAME
    };
    @Override
```

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Aclt = (AutoCompleteTextView)this.findViewById(R.id.AutoCompleteTextView1);
                                                     //获取对象
    tv = (TextView)this.findViewById(R.id.TextView1);
    ContentResolver content = getContentResolver();    //获取 ContentResolver 对象
    //取得通讯录的 Cursor
    cursor = content.query
    (
            Contacts.People.CONTENT_URI,
            PEOPLE_PROJECTION,
            null,
            null,
            Contacts.People.DEFAULT_SORT_ORDER
    );
    Ca = new ContactsAdapter(this,cursor);             //创建对象
    Aclt.setAdapter(Ca);
    Aclt.setOnItemClickListener
    (
        new AdapterView.OnItemClickListener()
        {
            public void onItemClick(AdapterView<?> arg0, View arg1,
                    int arg2, long arg3) {
                Cursor c = Ca.getCursor();             //取得 Cursor 对象
                c.moveToPosition(arg2);                //移动到单击位置
                 String number = c.getString(c.getColumnIndexOrThrow(Contacts.People.
NUMBER));                                              //获取电话号码
                if(number == null)
                {
                    number = "该联系人尚未有电话号码";
                }
                  String name = c.getString(c.getColumnIndexOrThrow(Contacts.People.
NAME));
        tv.setText("联系人姓名: " + name + ",联系人电话: " + number + ".");
            }
        }
    );
}
}
```

(4) 在 src\fs.li8_10search 包下创建一个 ContactsAdapter 类,在该类中获取通讯录中联系人的姓名,并获取搜索字符为“*”时的结果集。代码为:

```
package fs.li8_10search;
import android.content.ContentResolver;
import android.content.Context;
import android.database.Cursor;
import android.provider.Contacts;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;
public class ContactsAdapter extends CursorAdapter{
```

```
    ContentResolver Cr;
    public ContactsAdapter(Context context, Cursor c) {
        super(context, c);
        Cr = context.getContentResolver();
    }
    @Override
    public void bindView(View arg0, Context arg1, Cursor arg2) {
        //获取通讯录中的姓名
        ((TextView) arg0).setText(arg2.getString
(arg2.getColumnIndexOrThrow(Contacts.People.NAME)));
    }
    @Override
    public View newView(Context arg0, Cursor arg1, ViewGroup arg2)
{
        final LayoutInflater myLi = LayoutInflater.from(arg0);
        final TextView tv = (TextView)myLi.inflate
        (
android.R.layout.simple_dropdown_item_1line,arg2,false);
        tv.setText//设置显示文字
        (
            arg1.getString(arg1.getColumnIndexOrThrow
(Contacts.People.NAME))
        );
        return tv;
    }
    @Override
    public String convertToString(Cursor cursor)
    {
        String str = cursor.getString
(cursor.getColumnIndexOrThrow(Contacts.People.NAME));
        return str;
    }
    @Override
    public Cursor runQueryOnBackgroundThread(CharSequence cs)
    {
        if(getFilterQueryProvider()!= null)
        {
            return getFilterQueryProvider().runQuery(cs);
        }
        StringBuilder sb = new StringBuilder();
        String[] str = null;
        if(cs!= null)
        {
            sb.append("UPPER(");
            sb.append(Contacts.People.NAME);
            sb.append(") GLOB ?");
            str = new String[]
            {
                    cs.toString().toUpperCase() + "*"
            };
        }
        //返回搜索结果
        return Cr.query(
                Contacts.People.CONTENT_URI,
```

```
                    MainActivity.PEOPLE_PROJECTION,
                    sb == null?null:sb.toString(),
                    str,
                    Contacts.People.DEFAULT_SORT_ORDER
                    );
        }
    }
```

(5) 打开 AndroidManifest.xml 文件,设置通讯录搜索权限。代码为:

```
…
   </application>
<!-- 添加通讯录搜索权限 -->
     < uses - permission android: name = " android. permission. READ _ CONTACTS" > </uses -
permission>
</manifest>
```

运行程序,效果如图 8-16 所示。



图 8-16 通讯录搜索界面

8.7.7 震动功能

使用手机的震动函数针对 Notification 让手机执行特定样式的震动。Android 允许用户控制震动的样式,可以使用震动来传达信息以获取注意。为了设置震动样式,给 Notification 的 vibrate 属性设定一个时间数组,每个间隔的数字相应地代表震动或暂停的时间长度。

【例 8-11】 实现手机的震动功能。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_11Shock。

(2) 打开 res\layout 目录下的 main.xml 文件，文件中声明 4 个 TextView 控件及 4 个 ToggleButton 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< AbsoluteLayout xmlns:android = "http://schemas.android.com/apk/res/android"
  android:orientation = "vertical"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent"
  android:background = "#aaaccc" >
  < TextView
  android:id = "@ + id/myTextView1"
  android:layout_width = "fill_parent"
  android:layout_height = "wrap_content" />
  < TextView
  android:id = "@ + id/myTextView2"
  android:layout_width = "127px"
  android:layout_height = "35px"
  android:layout_x = "90px"
  android:layout_y = "33px"
  android:text = "短时震动"/>
  < TextView
  android:id = "@ + id/myTextView3"
  android:layout_width = "127px"
  android:layout_height = "35px"
  android:layout_x = "90px"
  android:layout_y = "115px"
  android:text = "长时震动"/>
  < TextView
  android:id = "@ + id/myTextView4"
  android:layout_width = "127px"
  android:layout_height = "35px"
  android:layout_x = "90px"
  android:layout_y = "216px"
  android:text = "节奏震动"/>
  < ToggleButton
  android:id = "@ + id/myTogglebutton1"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_x = "29px"
  android:layout_y = "31px" />
  < ToggleButton
  android:id = "@ + id/myTogglebutton2"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_x = "29px"
  android:layout_y = "114px"/>
  < ToggleButton
  android:id = "@ + id/myTogglebutton3"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_x = "29px"
  android:layout_y = "214px"/>
</AbsoluteLayout >
```

(3) 打开 src\fs.li8_11shock 包下的 MainActivity.java 文件，在文件中实现手机的震

动功能。代码为：

```
package fs.li8_11shock;
import android.app.Activity;
import android.app.Service;
import android.os.Bundle;
import android.os.Vibrator;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;
import android.widget.ToggleButton;
public class MainActivity extends Activity
{
  private Vibrator mVibrator01;
  /* 第一次调用 Activity 活动 */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /* 设置 ToggleButton 的对象 */
    mVibrator01 = ( Vibrator )getApplication().getSystemService
    (Service.VIBRATOR_SERVICE);
    final ToggleButton mtogglebutton1 =
    (ToggleButton) findViewById(R.id.myTogglebutton1);
    final ToggleButton mtogglebutton2 =
    (ToggleButton) findViewById(R.id.myTogglebutton2);
    final ToggleButton mtogglebutton3 =
    (ToggleButton) findViewById(R.id.myTogglebutton3);
    /* 短震动 */
    mtogglebutton1.setOnClickListener(new OnClickListener()
    {
      public void onClick(View v)
      {
        if (mtogglebutton1.isChecked())
        {
          /* 设置震动的周期 */
          mVibrator01.vibrate( new long[]{100,10,100,1000},-1);
          /* 用 Toast 显示震动启动 */
          Toast.makeText
          (
            MainActivity.this,
            getString(R.string.str_ok),
            Toast.LENGTH_SHORT
          ).show();
        }
        else
        {
          /* 取消震动 */
          mVibrator01.cancel();
          /* 用 Toast 显示震动已被取消 */
          Toast.makeText
          (
            MainActivity.this,
            getString(R.string.str_end),
            Toast.LENGTH_SHORT
```

```
        ).show();
      }
    }
  });
  /* 长震动 */
  mtogglebutton2.setOnClickListener(new OnClickListener()
  {
    public void onClick(View v)
    {
      if (mtogglebutton2.isChecked())
      {
        /* 设置震动的周期 */
        mVibrator01.vibrate(new long[]{100,100,100,1000},0);
        /* 用 Toast 显示震动启动 */
        Toast.makeText
        (
          MainActivity.this,
          getString(R.string.str_ok),
          Toast.LENGTH_SHORT
        ).show();
      }
      else
      {
        /* 取消震动 */
        mVibrator01.cancel();
        /* 用 Toast 显示震动取消 */
        Toast.makeText
        (
          MainActivity.this,
          getString(R.string.str_end),
          Toast.LENGTH_SHORT
        ).show();
      }
    }
  });
  /* 节奏震动 */
  mtogglebutton3.setOnClickListener(new OnClickListener()
  {
    public void onClick(View v)
    {
      if (mtogglebutton3.isChecked())
      {
        /* 设置震动的周期 */
        mVibrator01.vibrate( new long[]{1000,50,1000,50,1000},0);
        /* 用 Toast 显示震动启动 */
        Toast.makeText
        (
          MainActivity.this, getString(R.string.str_ok),
          Toast.LENGTH_SHORT
        ).show();
      }
      else
      {
        /* 取消震动 */
        mVibrator01.cancel();
        /* 用 Toast 显示震动取消 */
        Toast.makeText
```

```
                (
                  MainActivity.this,
                  getString(R.string.str_end),
                  Toast.LENGTH_SHORT
                ).show();
              }
            }
          });
        }
      }
```

(4) 打开 res\values 目录下的 strings.xml 文件,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name">震动功能</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
     < string name = "str_ok">震动中....</string >
      < string name = "str_end">震动结束</string >
</resources >
```

(5) 打开 AndroidManifest.xml 文件,设置手机震动权限。代码为:

```
…
    </application >
  < users - permission android:name = "android.permission.VIBRATE"/>
</manifest >
```

运行程序,效果如图 8-17 所示。



图 8-17　震动界面

8.7.8 WiFi 功能

WiFi 是一种可以将个人计算机、手持设备(如 PDA、手机)等终端以无线方式互相连接的技术。WiFi 是一个无线网络通信技术的品牌,由 WiFi 联盟(WiFi Alliance)所持有,目的是改善基于 IEEE 802.11 标准的无线网络产品之间的互通性。现在,有些人会把 WiFi 和 IEEE 802.11 混为一谈,甚至把 WiFi 等同于无线网际网络。

WiFi 是一种短程无线传输技术,能够在数百英尺范围内支持互联网接入的无线电信号。随着技术的发展,以及 IEEE 802.11a 和 IEEE 802.11g 等标准的出现,现在 IEEE 802.11 这个标准已被统称作 WiFi。从应用层面来说,要想使用 WiFi,用户首先要有 WiFi 兼容的用户端装置。

【例 8-12】 实现 WiFi 的打开与关闭功能。

其实现步骤如下:

(1) 在 Eclipse 环境下建立一个 Android 应用项目,命名为 li8_12WiFi。

(2) 打开 res\layout 目录下的 main.xml 文件,将代码修改为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
  xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent">
  < TextView
    android:id = "@ + id/myTextView1"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/hello_world"/>
  < CheckBox
    android:id = "@ + id/myCheckBox1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "@string/str_checked"/>
</LinearLayout >
```

(3) 打开 res\values 目录下的 strings.xml 文件,添加以下代码:

```
< string name = "str_checked">打开 WiFi </string >
< string name = "str_uncheck">关闭 WiFi </string >
< string name = "str_start_wifi_failed">打开失败</string >
< string name = "str_start_wifi_done">打开成功</string >
< string name = "str_stop_wifi_failed">打开失败</string >
< string name = "str_stop_wifi_done">关闭成功</string >
< string name = "str_wifi_enabling">正在启动...</string >
< string name = "str_wifi_disabling">正在关闭...</string >
< string name = "str_wifi_disabled">已关闭</string >
< string name = "str_wifi_unknow">未知...</string >
```

(4) 打开 src\fs.li8_12wifi 包下的 MainActivity 文件,将代码修改为:

```
public class MainActivity extends Activity
{
```

```
    private TextView mTextView1;
    private CheckBox mCheckBox1;
    /* 创建 WiFiManager 对象 */
    private WifiManager mWiFiManager1;
  //定义 mTextView1 和 mCheckBox1,分别用于显示提示文本和获取复选框的选择状态
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mTextView1 = (TextView) findViewById(R.id.myTextView1);
      mCheckBox1 = (CheckBox) findViewById(R.id.myCheckBox1);
      /* 以 getSystemServices 取得 WIFI_SERVICE,然后通过 if 语句来判断运行
       * 程序后的 WiFi 状态是否打开或正在打开中,这样即可显示对应的提示信息
       */
      mWiFiManager1 = (WifiManager) this.getSystemService(Context.WIFI_SERVICE);
      /* 判断运行程序后的 WiFi 状态是否打开或正在打开中 */
      if(mWiFiManager1.isWifiEnabled())
      {
        /* 判断 WiFi 状态是否"已打开" */
        if(mWiFiManager1.getWifiState() == WifiManager.WIFI_STATE_ENABLED)
        {
          /* 若 WiFi 已打开,将复选框选中 */
          mCheckBox1.setChecked(true);
          /* 更改文字为"关闭 WiFi" */
          mCheckBox1.setText(R.string.str_uncheck);
        }
        else
        {
          /* 若 WiFi 未打开,将复选框取消选中 */
          mCheckBox1.setChecked(false);
          /* 更改文字为"打开 WiFi" */
          mCheckBox1.setText(R.string.str_checked);
        }
      }
      else
      {
        mCheckBox1.setChecked(false);
        mCheckBox1.setText(R.string.str_checked);
      }
      /* 捕捉 CheckBox 的单击事件 */
      mCheckBox1.setOnClickListener(
      new CheckBox.OnClickListener()
      {
        @Override
        public void onClick(View v)
        {
          // TODO 自动生成的方法存根

          /* 当复选框为取消选中状态时 */
          if(mCheckBox1.isChecked() == false)
          {
            /* 尝试关闭 WiFi 服务 */
```

```
try
{
  /* 判断 WiFi 状态是否为已打开 */
  if(mWiFiManager1.isWifiEnabled() )
  {
    /* 关闭 WiFi */
    if(mWiFiManager1.setWifiEnabled(false))
    {
      mTextView1.setText(R.string.str_stop_wifi_done);
    }
    else
    {
      mTextView1.setText(R.string.str_stop_wifi_failed);
    }
  }
  else
  {
    /* WiFi 状态不为已打开状态时 */
    switch(mWiFiManager1.getWifiState())
    {
      /* WiFi 正在打开过程中,导致无法关闭 */
      case WifiManager.WIFI_STATE_ENABLING:
        mTextView1.setText
        (
          getResources().getText
          (R.string.str_stop_wifi_failed) + ":" +
          getResources().getText
          (R.string.str_wifi_enabling)
        );
        break;
      /* WiFi 正在关闭过程中,导致无法关闭 */
      case WifiManager.WIFI_STATE_DISABLING:
        mTextView1.setText
        (
          getResources().getText
          (R.string.str_stop_wifi_failed) + ":" +
          getResources().getText
          (R.string.str_wifi_disabling)
        );
        break;
      /* WiFi 已经关闭 */
      case WifiManager.WIFI_STATE_DISABLED:
        mTextView1.setText
        (
          getResources().getText
          (R.string.str_stop_wifi_failed) + ":" +
          getResources().getText
          (R.string.str_wifi_disabled)
        );
        break;
      /* 无法取得或辨识 WiFi 状态 */
      case WifiManager.WIFI_STATE_UNKNOWN:
      default:
        mTextView1.setText
```

```
                    (
                      getResources().getText
                      (R.string.str_stop_wifi_failed) + ":" +
                      getResources().getText
                      (R.string.str_wifi_unknow)
                    );
                    break;
                }
                mCheckBox1.setText(R.string.str_checked);
              }
            }
            catch (Exception e)
            {
              Log.i("HIPPO", e.toString());
              e.printStackTrace();
            }
          }
          else if(mCheckBox1.isChecked() == true)
          {
            /* 尝试打开 WiFi 服务 */
            try
            {
              /* 确认 WiFi 服务关闭且不在打开作业中 */
              if(!mWiFiManager1.isWifiEnabled() &&
                 mWiFiManager1.getWifiState()!=
                 WifiManager.WIFI_STATE_ENABLING )
              {
                if(mWiFiManager1.setWifiEnabled(true))
                {
                  switch(mWiFiManager1.getWifiState())
                  {
                    /* WiFi 正在打开过程中,导致无法打开 */
                    case WifiManager.WIFI_STATE_ENABLING:
                      mTextView1.setText
                      (
                        getResources().getText
                        (R.string.str_wifi_enabling)
                      );
                      break;
                    /* WiFi 已经打开,无法再次打开 */
                    case WifiManager.WIFI_STATE_ENABLED:
                      mTextView1.setText
                      (
                        getResources().getText
                        (R.string.str_start_wifi_done)
                      );
                      break;
                    /* 其他未知的错误 */
                    default:
                      mTextView1.setText
                      (
                        getResources().getText
                        (R.string.str_start_wifi_failed) + ":" +
                        getResources().getText
```

```
          (R.string.str_wifi_unknow)
        );
        break;
    }
  }
  else
  {
    mTextView1.setText(R.string.str_start_wifi_failed);
  }
}
else
{
  switch(mWiFiManager1.getWifiState())
  {
    /* WiFi正在打开过程中,导致无法打开 */
    case WifiManager.WIFI_STATE_ENABLING:
      mTextView1.setText
      (
        getResources().getText
        (R.string.str_start_wifi_failed) + ":" +
        getResources().getText
        (R.string.str_wifi_enabling)
      );
      break;
    /* WiFi正在关闭过程中,导致无法打开 */
    case WifiManager.WIFI_STATE_DISABLING:
      mTextView1.setText
      (
        getResources().getText
        (R.string.str_start_wifi_failed) + ":" +
        getResources().getText
        (R.string.str_wifi_disabling)
      );
      break;
    /* WiFi已经关闭 */
    case WifiManager.WIFI_STATE_DISABLED:
      mTextView1.setText
      (
        getResources().getText
        (R.string.str_start_wifi_failed) + ":" +
        getResources().getText
        (R.string.str_wifi_disabled)
      );
      break;
    /* 无法取得或识别WiFi状态 */
    case WifiManager.WIFI_STATE_UNKNOWN:
    default:
      mTextView1.setText
      (
        getResources().getText
        (R.string.str_start_wifi_failed) + ":" +
        getResources().getText
        (R.string.str_wifi_unknow)
      );
```

```
                            break;
                        }
                    }
                    mCheckBox1.setText(R.string.str_uncheck);
                }
                catch (Exception e)
                {
                    Log.i("HIPPO", e.toString());
                    e.printStackTrace();
                }
            }
        }
    });
}
//定义 mMakeTextToast()方法,根据当前操作显示对应的提示性信息
public void mMakeTextToast(String str, boolean isLong)
{
    if(isLong == true)
    {
        Toast.makeText(MainActivity.this, str, Toast.LENGTH_LONG).show();
    }
    else
    {
        Toast.makeText(MainActivity.this, str, Toast.LENGTH_SHORT).show();
    }
}
@Override
protected void onResume()
{
    /* 在 onResume 中重写事件为取得所打开程序当下的 WiFi 状态 */
    try
    {
        switch(mWiFiManager1.getWifiState())
        {
            /* WiFi 处于已经打开状态 */
            case WifiManager.WIFI_STATE_ENABLED:
                mTextView1.setText
                (
                    getResources().getText(R.string.str_wifi_enabling)
                );
                break;
            /* WiFi 处于正在打开状态 */
            case WifiManager.WIFI_STATE_ENABLING:
                mTextView1.setText
                (
                    getResources().getText(R.string.str_wifi_enabling)
                );
                break;
            /* WiFi 正在关闭过程中 */
            case WifiManager.WIFI_STATE_DISABLING:
                mTextView1.setText
                (
                    getResources().getText(R.string.str_wifi_disabling)
                );
```

```
                break;
            /* WiFi 已经关闭 */
            case WifiManager.WIFI_STATE_DISABLED:
                mTextView1.setText
                (
                  getResources().getText(R.string.str_wifi_disabled)
                );
                break;
            /* 无法取得或识别 WiFi 状态 */
            case WifiManager.WIFI_STATE_UNKNOWN:
            default:
                mTextView1.setText
                (
                  getResources().getText(R.string.str_wifi_unknow)
                );
                break;
          }
        }
        catch(Exception e)
        {
          mTextView1.setText(e.toString());
          e.getStackTrace();
        }
        super.onResume();
      }
      @Override
      protected void onPause()
      {
        super.onPause();
      }
    }
```

(5) 授予权限。打开 AndroidManifest.xml 文件,代码为:

```
  …
  </application>
  <!-- 声明 WiFi 以及网络等相关权限 -->
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
 <uses-permission android:name="android.permission.WAKE
_LOCK"/>
  </manifest>
```

运行程序,效果如图 8-18 所示。当选择复选框后会执行对应的操作,并显示对应的提示信息。

图 8-18　WiFi 页面

8.7.9　手机桌面设置

每个人都会有自己喜欢的相片,将自己喜欢的相片设置为手机的背景基本上成为每个手机用户的习惯,那么在 Android 中怎样实现手机桌面的设置呢?

下面通过一个简单的实例来演示如何实现 Android 手机桌面的设置。

【例 8-13】 本例界面由一个 TextView 和两个 Button 构成，当单击“更换桌面背景”按钮时，系统自动将已设定好的图片设置为手机的桌面背景，同时出现 Toast 提示用户桌面背景已经置换成功。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_13Desktop。

(2) 打开 res\layout 目录下的 main.xml 文件，代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:background = "#aaaccc"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    <TextView
        android:id = "@ + id/textView1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "请单击更换手机桌面背景" />
    <Button
        android:id = "@ + id/Button2"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/textView1"
        android:layout_marginTop = "116dp"
        android:text = "退出查看" />
    <Button
        android:id = "@ + id/Button1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/Button2"
        android:layout_below = "@ + id/textView1"
        android:layout_marginTop = "53dp"
        android:text = "更换桌面背景" />
</RelativeLayout>
```

(3) 打开 src\fs.li8_13desktop 包下的 MainActivity.java 文件，用于实现手机桌面的更换。代码为：

```
package fs.li8_13desktop;
import java.io.InputStream;
import android.app.Activity;
import android.content.res.Resources;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

```
import android.widget.Toast;
public class MainActivity extends Activity {
    InputStream is;
    Button button;
    Button exit;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        button = (Button)this.findViewById(R.id.Button1);
        exit = (Button)this.findViewById(R.id.Button2);
        button.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v) {
                Resources resource = getBaseContext().getResources();    //获取 resource
                        is = resource.openRawResource(R.drawable.bj2);  //加载图片
                        try
                        {
                            MainActivity.this.setWallpaper(is);         //设置桌面背景
                        }catch(Exception e)
                        {
                            e.printStackTrace();
                        }
                          Toast.makeText(MainActivity.this, "已经成功置换桌面背景!!",
Toast.LENGTH_SHORT).show();
                    }
                }
        );
        exit.setOnClickListener
        (
            new OnClickListener()
            {
                public void onClick(View v) {
                    System.exit(0);                         //退出程序
                }
            }
        );
    }
}
```

(4) 打开 AndroidManifest.xml 文件,设置更换桌面背景权限。代码为:

```
…
    </application>
    <!-- 设置权限 -->
     <uses-permission android:name="android.permission.SET_WALLPAPER"></uses-
permission>
</manifest>
```

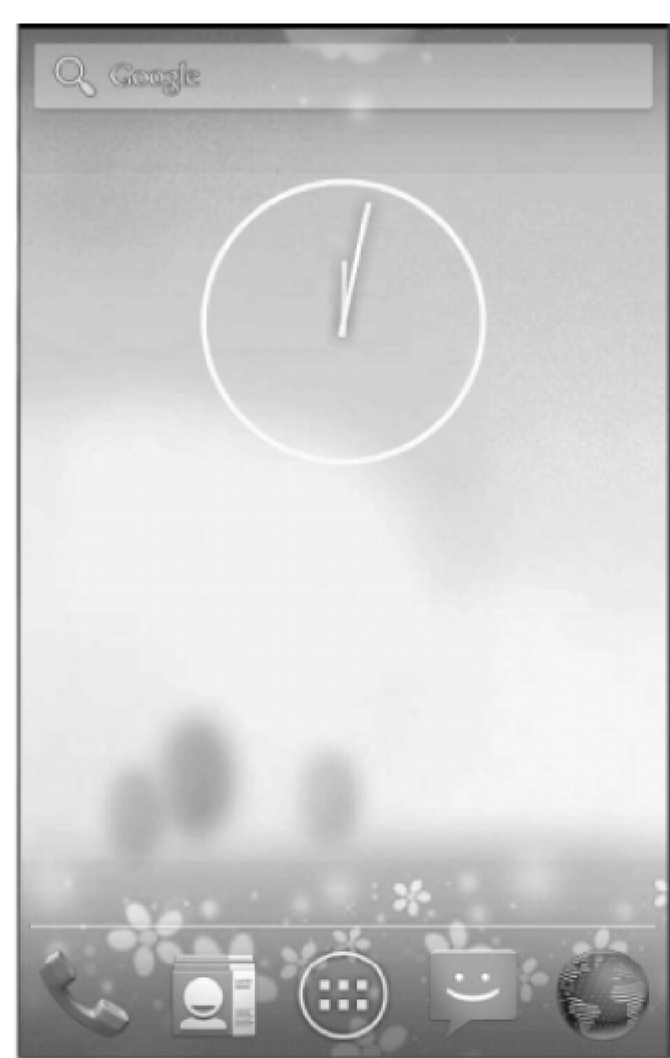
运行程序,效果如图 8-19 所示。



(a) 初始界面



(b) 单击“更换桌面背景”按钮



(c) 更换背景效果

图 8-19　更换手机桌面背景

8.8　综合实例

本节通过一个电话免扰实例来综合说明手机通信应用。所谓的电话免扰，即指定的电话号码呼入电话时，自动挂断该号码并回复短信。

【例 8-14】　电话免扰实例。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_14Interference。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中设置拦截号码输入框、回复短信内容数据框、“保存设置”按钮以及“开启拦截”按钮。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aaaccc">
    <TextView
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_centerHorizontal = "true"
        android:text = "@string/hello_world" />
    <EditText
```

```
        android:id = "@ + id/editText1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/textView1"
        android:layout_marginTop = "23dp"
        android:ems = "10"
        android:hint = "请输入拦截的号码" />
    < EditText
        android:id = "@ + id/edittext2"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/editText1"
        android:layout_marginTop = "35dp"
        android:ems = "10"
        android:hint = "请输入自动回复的短信内容"/>
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentBottom = "true"
        android:layout_alignRight = "@ + id/textView1"
        android:layout_marginRight = "49dp"
        android:text = "保存设置"/>
    < Button
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentBottom = "true"
        android:layout_toRightOf = "@ + id/button1"
        android:text = "开启拦截"/>
</RelativeLayout >
```

(3) 打开 src\fs. li8_14interference 包下的 MainActivity. java 文件，开发按钮逻辑，包括初始界面、保存设置以及拦截是否开启。代码为：

```
package fs. li8_14interference;
import android. os. Bundle;
import android. app. Activity;
import android. content. SharedPreferences;
import android. content. SharedPreferences. Editor;
import android. view. Menu;
import android. view. View;
import android. view. View. OnClickListener;
import android. widget. Button;
import android. widget. EditText;
import android. widget. Toast;
public class MainActivity extends Activity {
EditText et_phonenum, et_sms;
Button btn_save, btn_open;
SharedPreferences sp;
boolean is_open = false;
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        et_phonenum = (EditText)findViewById(R.id.editText1);
        et_sms = (EditText)findViewById(R.id.edittext2);
        btn_save = (Button)findViewById(R.id.button1);
        btn_open = (Button)findViewById(R.id.button2);
        //获得保存在 SharedPreferences 中的数据,包括电话号码、短信内容以及是否拦截,并在对
        //应的位置显示保存的内容
        sp = getSharedPreferences("SP", MODE_PRIVATE);
        String phoneString = sp.getString("phone", "");
        String smsString = sp.getString("sms", "");
        is_open = sp.getBoolean("open", false);
        if (!phoneString.equals("")) {
            et_phonenum.setText(phoneString);
        }
        if (!smsString.equals("")) {
            et_sms.setText(smsString);
        }
        if (is_open) {
            btn_open.setText("关闭拦截");
        }
    //添加"保存设置"按钮的监听事件,判断输入的内容,保存在 SharedPreferences 中
        btn_save.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO 自动生成的方法存根
                String phoneString = et_phonenum.getText().toString();
                String smsString = et_sms.getText().toString();
                if ((!phoneString.equals("")) &&(!smsString.equals(""))) {
                    Editor editor = sp.edit();
                    editor.putString("phone", phoneString);
                    editor.putString("sms", smsString);
                    editor.commit();
                    Toast.makeText(MainActivity.this, "已保存设置", Toast.LENGTH_LONG).
show();
                }else {
                    Toast.makeText(MainActivity.this, "请输入号码和短信内容", Toast.
LENGTH_LONG).show();
                }
            }
        });
        //添加拦截按钮的监听事件
        btn_open.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO 自动生成的方法存根
                if (is_open) {
                    is_open = false;
                    btn_open.setText("开启拦截");
                }else {
                    is_open = true;
```

```
                btn_open.setText("关闭拦截");
            }
            Editor editor = sp.edit();
            editor.putBoolean("open", is_open);
            editor.commit();
        }
    });
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // 增加项目操作栏
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}
```

(4) 在 src\fs.li8_14interference 包下创建一个名为 Reply.java 的文件,用于开发电话状态的广播处理,包括挂断电话及发送短信。代码为:

```
package fs.li8_14interference;
import java.net.ContentHandler;
import java.util.ArrayList;
import android.app.Service;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.telephony.SmsManager;
import android.telephony.TelephonyManager;
import android.widget.Toast;
public class Reply extends BroadcastReceiver {
    String phoneNumber, ed_num, ed_sms;
    boolean is_open;
    SharedPreferences sp;
    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO 自动生成的方法存根
        TelephonyManager tm = (TelephonyManager) context
            .getSystemService(Service.TELEPHONY_SERVICE);// 获取电话管理器
        switch (tm.getCallState()) {                 // 判断电话状态
        case TelephonyManager.CALL_STATE_RINGING:    // 来电响铃
            try {
                // 来电拒听
            phoneNumber = intent.getStringExtra("incoming_number");         // 获得号码
                //获得呼入号码以及是否拦截状态,如果不拦截则直接返回
                sp = context.getSharedPreferences("SP", Context.MODE_PRIVATE);
                is_open = sp.getBoolean("open", false);
                if (!is_open) {
                    break;
                }
                //当为拦截状态时,判断呼入号码是否为指定号码,挂断电话回复短信,对于挂断
                //电话,需要使用隐藏方法
                ed_num = sp.getString("phone", "");
    if (phoneNumber.equals(ed_num)) {                // 对比判断是否是拦截号码
```

```
                    Toast.makeText(context, "号码" + phoneNumber + "已经被挂断拦截",
                            1000).show();
                        // 发送短信
                        new Thread(new Runnable() {
                            @Override
                            public void run() {
                                // TODO 自动生成的方法存根
                                ed_sms = sp.getString("sms", "不方便接听电话");
                                sendSmS(ed_num, ed_sms);
                            }
                        }).start();
                    }
                } catch (Exception e) {
                }
                break;
            case TelephonyManager.CALL_STATE_OFFHOOK:        //来电接通知,电话拨出
                break;
            case TelephonyManager.CALL_STATE_IDLE:           //电话挂断
                break;
            }
        }
        //发送短信
        private void sendSmS(String ph_num, String message) {
            SmsManager sms = SmsManager.getDefault();
            if (message.length() > 70) {
                ArrayList<String> msgs = sms.divideMessage(message);
                for (String msg : msgs) {
                    sms.sendTextMessage(ph_num, null, msg, null, null);
                }
            } else {
                sms.sendTextMessage(ph_num, null, message, null, null);
            }
        }
    }
```

(5) 打开 AndroidManifest.xml 文件,用于添加广播的注册以及添加需要的权限。代码为:

```
...
 <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <!-- 改变电话状态 -->
    <uses-permission android:name="android.permission.MODEIFY_PHONE_STATE"/>
    <!-- 获取电话状态 -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <!-- 拨出电话 -->
    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
    <!-- 电话 -->
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <!-- 发送短信 -->
    <uses-permission android:name="android.permission.SEND_SMS"/>"
...
```

第 9 章　Android 手机自动控制服务

手机自动控制服务是 Android 手机自带的服务，其自动服务功能包含应用程序的后台运行、实时监控是否收到短信等。

9.1　查看手机信息

每一部手机都有其详细的信息，这些信息包括手机号码、电信网络国别等。

本例设计：在界面上有一个按钮，当单击该按钮时，即可获取手机上的相关信息，信息包括手机号码、电信网络国别、电信公司名称、手机 SIM 码、手机通信类型、手机网络类型、是否漫游以及蓝牙和 WiFi 的状态等。当单击其中的一条信息时会有 Toast 弹出，给出所单击的信息。

TelephonyManager 是一个管理手机通话状态、电话网络信息的服务类。

以下实现中获取 TelephonyManager 十分简单，主要通过 getSystemService 获取 TelephonyManager 对象，接着通过 TelephonyManager 的方法来获取和电信有关的网络信息，然后通过 Android. provider. Setting. System. getString()获取手机的相关设置信息，最后将 setListAdapter 内的信息显示在 ListView 中。

【例 9-1】　查询手机的相关信息。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li9_1Phone_Msg。

(2) 打开 res\layout 目录下的 main. xml 文件，代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aaaccc">
    <Button
         android:text = "查看手机信息"
         android:id = "@+id/Button01"
         android:layout_width = "fill_parent"
         android:layout_height = "wrap_content"/>
```

```
    <LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:background = "#ffcc66"
        android:paddingLeft = "0dip"
        android:paddingRight = "5dip"
        android:paddingTop = "5dip">
            <ListView
                android:id = "@+id/ListView01"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:cacheColorHint = "#aaaccc"/>
    </LinearLayout>
</RelativeLayout>
```

(3) 打开 src\fs.li9_1phone_msg 包下的 MainActivity.java 文件,用于实现手机信息查询。代码为:

```
public class MainActivity extends Activity {
    private ListView lv;
    private TelephonyManager tm;
    private ContentResolver cr;
    private List<String> list = new ArrayList<String>();
    private List<String> name = new ArrayList<String>();
    private Button bCheck;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        lv = (ListView)this.findViewById(R.id.ListView01);
        tm = (TelephonyManager)getSystemService(TELEPHONY_SERVICE);
        cr = MainActivity.this.getContentResolver();
        bCheck = (Button)this.findViewById(R.id.Button01);
        String str = null;                              //记录 cr 获取的信息
        name.add("手机号码: ");
        name.add("电信网络国别: ");
        name.add("电信公司代码: ");
        name.add("电信公司名称: ");
        name.add("SIM 码: ");
        name.add("手机通信类型: ");
        name.add("手机网络类型 : ");
        name.add("手机是否漫游: ");
        name.add("蓝牙状态: ");
        name.add("WiFi 状态: ");
        if(tm.getLine1Number()!= null)                  //手机号码
        {
            list.add(tm.getLine1Number());
        }else
        {
            list.add("无法取得您的电话号码");
        }
        if(!tm.getNetworkCountryIso().equals(""))       //电信网络国别
        {
```

```
            list.add(tm.getNetworkCountryIso());
        }else
        {
            list.add("无法取得您的电信网络国别");
        }
        if(!tm.getNetworkOperator().equals(""))        //电信公司代码
        {
            list.add(tm.getNetworkOperator());
        }else
        {
            list.add("无法获取电信公司代码");
        }
        if(!tm.getNetworkOperatorName().equals(""))    //电信公司名称
        {
            list.add(tm.getNetworkOperatorName());
        }else
        {
            list.add("无法获取电信公司名称");
        }
        if(tm.getSimSerialNumber()!= null)             //手机 SIM 码
        {
            list.add(tm.getSimSerialNumber());
        }else
        {
            list.add("无法获取手机 SIM 码");
        }
    if(tm.getPhoneType() == TelephonyManager.PHONE_TYPE_GSM)//手机通信类型
        {
            list.add("GSM");
        }
        else
        {
            list.add("无法获取手机通信类型");
        }
        if(tm.getNetworkType() == TelephonyManager.NETWORK_TYPE_EDGE)   //获取手机网络类型
        {
            list.add("EDGE");
        }else if(tm.getNetworkType() == TelephonyManager.NETWORK_TYPE_GPRS)
        {
            list.add("GPRS");
        }else if(tm.getNetworkType() == TelephonyManager.NETWORK_TYPE_UMTS)
        {
            list.add("UMTS");
        }
        else
        {
            list.add("无法获取手机网络类型");
        }
        if(tm.isNetworkRoaming())                      //手机是否漫游
        {
            list.add("手机漫游中");
```

```
            }else
            {
                list.add("手机无漫游");
            }
            str = android.provider.Settings.System.getString(
                    cr,android.provider.Settings.System.BLUETOOTH_ON
                    );
            if(str.equals("1"))
            {
                list.add("蓝牙已打开");
            }else
            {
                list.add("蓝牙未打开");
            }
             str = android.provider.Settings.System.getString(cr, android.provider.Settings.
System.WiFi_ON);
            if(str.equals("1"))
            {
                list.add("WiFi 已打开");
            }else
            {
                list.add("WiFi 未打开");
            }
            bCheck.setOnClickListener
            (
                    new OnClickListener()
                    {
                        public void onClick(View v) {
                            BaseAdapter ba = new BaseAdapter()    //创建适配器
                            {
                                public int getCount() {
                                    return list.size();
                                }
                                public Object getItem(int position) {
                                    return null;
                                }
                                public long getItemId(int position) {
                                    return 0;
                                }
                            public View getView(int arg0, View arg1, ViewGroup arg2) {
                            LinearLayout ll = new LinearLayout(MainActivity.this);
                                    ll.setOrientation(LinearLayout.HORIZONTAL);
                                    ll.setPadding(5, 5, 5, 5);
                    TextView tv = new TextView(MainActivity.this);   //初始化 TextView
                                    tv.setTextColor(Color.BLACK); //设置字体颜色
                                    tv.setPadding(5,5,5,5);
                                    tv.setText(name.get(arg0));   //添加任务名字
                                    tv.setGravity(Gravity.LEFT);  //左对齐
                                    tv.setTextSize(18);           //字体大小
                            ll.addView(tv);                      //LinearLayout 添加 TextView
TextView tvv = new TextView(MainActivity.this);                      //初始化 TextView
```

```
                        tvv.setTextColor(Color.BLACK);//设置字体颜色
                        tvv.setPadding(5,5,5,5);
                        tvv.setText(list.get(arg0));  //添加任务名字
                        tvv.setGravity(Gravity.LEFT); //左对齐
                        tvv.setTextSize(18);          //字体大小
                        ll.addView(tvv);             //LinearLayout 添加 TextView
                        return ll;
                    }
                };

    lv.setAdapter(ba);                                //设置适配器
                lv.setOnItemClickListener           //设置选中菜单的监听器
                (
                    new OnItemClickListener()
                    {
                public void onItemClick(AdapterView<?> arg0, View arg1,
                                     int arg2, long arg3) {
                                Toast.makeText(MainActivity.this, name.get(arg2)
+ "" + list.get(arg2), Toast.LENGTH_SHORT).show();
                            }
                    }
                );
            }
        }
    );
  }
}
```

运行程序，效果如图 9-1(a)所示，单击界面中的“查看手机信息”按钮，效果如图 9-1(b)所示；单击任何一项信息，即显示对应的 Toast 提示，效果如图 9-1(c)所示。



(a) 默认界面　　(b) 相关信息显示　　(c) Toast提示

图 9-1　显示手机相关信息

9.2 查看 SIM 信息

手机的 SIM 卡是手机的一个重要的组成部分，没有 SIM 卡就不能正常拨打电话，本节介绍怎样获取 SIM 卡的信息。

该例通过使用 TelephonyManager 获取手机 SIM 卡的相关信息，并将获得的 SIM 卡状态、卡号、SIM 卡供应商、SIM 卡供应商名称、SIM 卡国别以 ListView 形式呈现在界面上。使用 TelephonyManager 获取手机 SIM 卡的信息需要为手机添加权限声明，权限代码为“<uses-permission android:name="android.permission.READ_PHONE_STATE"/>”。

【例 9-2】 实现查看 SIM 信息。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li9_2SIM_Msg2。

(2) 打开 res\layout 目录下的 main.xml 文件，代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aaaccc" >
<Button
         android:text = "查看 SIM 卡信息"
         android:id = "@ + id/Button1"
         android:layout_width = "fill_parent"
         android:layout_height = "wrap_content"/>
    <LinearLayout
         android:orientation = "vertical"
         android:layout_width = "fill_parent"
         android:layout_height = "wrap_content"
         android:background = "#ffcc66"
         android:paddingLeft = "0dip"
         android:paddingRight = "5dip"
         android:paddingTop = "5dip">
             <ListView
                 android:id = "@ + id/ListView1"
                 android:layout_width = "wrap_content"
                 android:layout_height = "wrap_content"
                 android:cacheColorHint = "#ffcc66"/>
    </LinearLayout>
</RelativeLayout>
```

(3) 打开 src\fs.li9_2sim_msg2 包下的 MainActivity.java 文件，实现 SIM 信息查询。代码为：

```
public class MainActivity extends Activity {
```

```java
    private ListView lv;
    private TelephonyManager tm;
    private List<String> list = new ArrayList<String>();
    private List<String> name = new ArrayList<String>();
    private Button bCheck;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        lv = (ListView)this.findViewById(R.id.ListView1);
        tm = (TelephonyManager)getSystemService(TELEPHONY_SERVICE);
        bCheck = (Button)this.findViewById(R.id.Button1);
        name.add("SIM 卡的状态: ");
        name.add("SIM 卡号: ");
        name.add("SIM 卡供应商号: ");
        name.add("SIM 卡供应商名称: ");
        name.add("SIM 卡国别: ");
        //通过 TelephonyManager 对象判断 SIM 卡状态、SIM 卡卡号、SIM 卡供应商代号、供应商名称
以及 SIM 卡国别,最后将获取的信息添加到 list 列表中
        if(tm.getSimState() == TelephonyManager.SIM_STATE_READY)//SIM 卡状态
        {
            list.add("状态良好");
        }else if(tm.getSimState() == TelephonyManager.SIM_STATE_ABSENT)
        {
            list.add("您目前没有 SIM 卡");
        }else if(tm.getSimState() == TelephonyManager.SIM_STATE_UNKNOWN)
        {
            list.add("SIM 卡处于未知状态");
        }
        if(tm.getSimSerialNumber()!= null)                //SIM 卡卡号
        {
            list.add(tm.getSimSerialNumber());
        }else
        {
            list.add("没有 SIM 卡卡号");
        }
        if(!tm.getSimOperator().equals(""))              //SIM 卡供应商代号
        {
            list.add(tm.getSimOperator());
        }else
        {
            list.add("没有 SIM 卡供应商代号");
        }
        if(!tm.getSimOperatorName().equals(""))          //SIM 卡供应商名称
        {
            list.add(tm.getSimOperatorName());
        }else
        {
            list.add("没有 SIM 卡供应商名称");
        }
        if(!tm.getSimCountryIso().equals(""))
        {
            list.add(tm.getSimCountryIso());
        }else
```

```
        {
            list.add("无法获取 SIM 国别");
        }
        bCheck.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v) {
                        BaseAdapter ba = new BaseAdapter()          //创建适配器
                    {
                            public int getCount() {
                                return list.size();
                            }
                            public Object getItem(int position) {
                                return null;
                            }
                            public long getItemId(int position) {
                                return 0;
                            }
                        public View getView(int arg0, View arg1, ViewGroup arg2) {
                        LinearLayout ll = new LinearLayout(MainActivity.this);
                                ll.setOrientation(LinearLayout.HORIZONTAL);
                                ll.setPadding(5, 5, 5, 5);
                    TextView tv = new TextView(MainActivity.this); //初始化 TextView
                                tv.setTextColor(Color.BLACK);   //设置字体颜色
                                tv.setPadding(5,5,5,5);
                                tv.setText(name.get(arg0));     //添加任务名字
                                tv.setGravity(Gravity.LEFT);    //左对齐
                                tv.setTextSize(18);             //字体大小
                                ll.addView(tv);               //LinearLayout 添加 TextView
                TextView tvv = new TextView(MainActivity.this);     //初始化 TextView
                                tvv.setTextColor(Color.BLACK);  //设置字体颜色
                                tvv.setPadding(5,5,5,5);
                                tvv.setText(list.get(arg0));    //添加任务名字
                                tvv.setGravity(Gravity.LEFT);   //左对齐
                                tvv.setTextSize(18);            //字体大小
                                ll.addView(tvv);              //LinearLayout 添加 TextView
                                return ll;
                            }
                    };
                    lv.setAdapter(ba);                          //设置适配器
                    lv.setOnItemClickListener                   //设置选中菜单的监听器
                    (
                                new OnItemClickListener()
                                {
                        public void onItemClick(AdapterView<?> arg0, View arg1,
                                        int arg2, long arg3) {
                                    Toast.makeText(MainActivity.this, name.get(arg2)
+ "" + list.get(arg2), Toast.LENGTH_SHORT).show();
                                }
                            }
                    );
                  }
                });
    }
```

(4) 打开 AndroidManifest.xml 文件，为文件添加权限。代码为：

```
…
    </application>
<!-- 添加权限 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
</manifest>
```

运行程序，效果如图 9-2(a)所示。当单击界面中的"查看 SIM 卡信息"按钮时，效果如图 9-2(b)所示，当单击图 9-2(b)中的任何一项时弹出相应的 Toast 说明，如图 9-2(c)所示。



图 9-2 显示 SIM 相关信息

9.3 闹钟设置

在 Android 中可以通过 AlarmManager 实现闹钟，AlarmManager 类专门用来设定在某个指定的时间去完成指定的事件。AlarmManager 提供了访问系统警报的服务，只要在程序中设置了警报服务，AlarmManager 就会通过 onReceive()方法执行这些事件，即使系统处于待机状态，也不会影响运行，可以通过 Context.getSystemService 方法来获得该服务。

AlarmManager 不仅可用于开发闹钟应用，还可作为一个全局定时器使用，在 Android 的程序中也是通过 Context 的 getSystemService()方法来获取 AlarmManager 对象的，一旦程序获取了 AlarmManager 对象，就可以调用它的以下方法来设置定时启动指定组件。

(1) void set(int type, long triggerAtTime, PendingIntent operation)：设置在 triggerAtTime 时间启动由 operation 参数指定的组件。其中，第一个参数指定定时服务的类型，该参数可接受以下值。

- ELAPSED_REALTIME：指定从现在开始时间过了一定时间后启动 operation 所

对应的组件。

- ELAPSED_REALTIME_WAKEUP：指定从现在开始时间过了一定时间后启动 operation 所对应的组件，即使系统关机也会执行 operation 所对应的组件。
- RTC：指定当系统调用 System.currentTimeMillis()方法返回的值与 triggerAtTime 相等时启动 operation 所对应的组件。
- RTC_WAKEUP：指定当系统调用 System.currentTimeMillis()方法返回的值与 triggerAtTime 相等时启动 operation 所对应的组件，即使系统关机也会执行 operation 所对应的组件。

(2) void setInexactRepeating(int type, long triggerAtTime, long interval, PendingIntent operation)：设置一个非精确的周期性任务。

(3) void setRepeating(int type, long triggerAtTime, long interval, PendingIntent operation)：设置一个周期性执行的定时服务。

(4) void cance(PendingIntent operation)：取消 AlarmManager 的定时服务。

下面通过一个实例来演示手机闹钟的设置。

【例 9-3】 在 Android 中实现闹钟设置。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li9_3Clock。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 TextView 控件、两个 Button 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aaaccc" >
    <TextView android:layout_width = "fill_parent"
        android:id = "@ + id/TextView"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
    <Button
        android:id = "@ + id/Button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentRight = "true"
        android:layout_below = "@ + id/Button1"
        android:layout_marginTop = "66dp"
        android:text = "取消闹钟" />
    <Button
        android:id = "@ + id/Button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
```

```
        android:layout_alignLeft = "@ + id/Button2"
        android:layout_alignParentRight = "true"
        android:layout_below = "@ + id/TextView"
        android:layout_marginTop = "47dp"
        android:text = "设置闹钟" />
</RelativeLayout>
```

(3) 打开 src\fs. li9_3clock 包下的 MainActivity. java 文件,在文件中实现闹钟的设置。代码为:

```
package fs.li9_3clock;
import java.util.Calendar;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.app.TimePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.TimePicker;
public class MainActivity extends Activity {
    /* 第一次调用 Activity 活动 */
    private TextView tv = null;
    private Button btn_set = null;
    private Button btn_cel = null;
    private Calendar c = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        tv = (TextView) this.findViewById(R.id.TextView);
        btn_set = (Button) this.findViewById(R.id.Button1);
        btn_cel = (Button) this.findViewById(R.id.Button2);
        c = Calendar.getInstance();
        btn_set.setOnClickListener(new Button.OnClickListener(){
            public void onClick(View v) {
                // TODO 自动生成的方法存根
                c.setTimeInMillis(System.currentTimeMillis());
                int hour = c.get(Calendar.HOUR_OF_DAY);
                int minute = c.get(Calendar.MINUTE);
                new TimePickerDialog(MainActivity.this,new TimePickerDialog.OnTimeSetListener(){
                    public void onTimeSet(TimePicker view, int hourOfDay,
                            int minute) {
                        // TODO 自动生成的方法存根
                        c.setTimeInMillis(System.currentTimeMillis());
                        c.set(Calendar.HOUR_OF_DAY, hourOfDay);
                        c.set(Calendar.MINUTE, minute);
                        c.set(Calendar.SECOND, 0);
                        c.set(Calendar.MILLISECOND, 0);
                  Intent intent = new Intent(MainActivity.this,AlamrReceiver.class);
```

```
            PendingIntent pi = PendingIntent.getBroadcast(MainActivity.this, 0, intent, 0);
    AlarmManager am = (AlarmManager) getSystemService(Activity.ALARM_SERVICE);
                    am.set(AlarmManager.RTC_WAKEUP, c.getTimeInMillis(), pi);//设置闹钟
                            am.setRepeating(AlarmManager.RTC_WAKEUP, c.getTimeInMillis(), (10
* 1000), pi);                                                   //重复设置
                            tv.setText("设置的闹钟时间为: " + hourOfDay + ":" + minute);
                        }
                    },hour,minute,true).show();
                }
            });
            btn_cel.setOnClickListener(new Button.OnClickListener(){
                public void onClick(View v) {
                    // TODO 自动生成的方法存根
                    Intent intent = new Intent(MainActivity.this,AlamrReceiver.class);
            PendingIntent pi = PendingIntent.getBroadcast(MainActivity.this, 0, intent, 0);
                    AlarmManager am = (AlarmManager) getSystemService(Activity.ALARM_SERVICE);
                    am.cancel(pi);
                    tv.setText("闹钟取消");
                }
            });
        }
    }
```

(4) 在 src\fs.li9_3clock 包下创建一个 AlamrReceiver.java 广播类,代码为:

```
package fs.li9_3clock;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;
public class AlamrReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO 自动生成的方法存根
        Toast.makeText(context, "闹钟时间到", Toast.LENGTH_LONG).show();
    }
}
```

(5) 打开 AndroidManifest.xml 文件,在文件中添加闹钟权限。代码为:

```
…
        </activity>
          <!-- 设置闹钟权限 -->
   <receiver android:name = ".AlamrReceiver" android:process = ":remote"></receiver>
    </application>
</manifest>
```

运行程序,效果如图 9-3(a)所示。单击界面中的"设置闹钟"按钮,弹出时间设置闹钟,效果如图 9-3(b)所示,时间设置完后单击 Done 按钮,则设置的时间显示在 TextView 中,并弹出对应的 Toast 提示,效果如图 9-3(c)所示。



(a) 默认界面　　(b) 设置闹钟时间　　(c) 显示闹钟时间

图 9-3　手机闹钟设置

9.4　查看电池剩余量

手机在使用过程中,最让人担心的是因没电而影响联系和业务,所以及时显示电池容量是非常有必要的。

对于这一功能,可以使用 Android API 中的 BroadcastReseiver 类和 Button 的 Listener 类实现,当 Reseiver 被注册后会在后台等待被其他程序调用;当指定要捕捉的 Action 发生时,Reseiver 就会被调用,并运行 onReseiver 来实现里面的程序。

下面通过一个实例来实现在 Android 手机中查看电池剩余量。

【例 9-4】 在实例中将利用 BroadcastReseiver 的特性获取手机电池的容量,即通过注册 BroadcastReseiver 时设置的 IntentFiler 来获取系统发出的 Intent.ACTION_BATTERY_CHANGED,然后以此获取电池的容量。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li9_4Capacity。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 TextView 控件和一个 Button 控件。代码为:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
```

```
      android:background = "#aaaccc">
    <TextView
      android:id = "@ + id/myTextView1"
      android:layout_width = "fill_parent"
      android:layout_height = "wrap_content"
      android:textSize = "20sp"
      android:text = "获取电池容量"
      android:layout_x = "60px"
      android:layout_y = "40px"/>
    <Button
        android:id = "@ + id/myButton1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/myTextView1"
        android:layout_below = "@ + id/myTextView1"
        android:layout_marginTop = "64dp"
        android:text = "获取"
        android:textSize = "14sp" />
</RelativeLayout>
```

(3) 在 res\layout 目录下创建一个 exadialog. xml 文件,在文件中声明一个 TextView 控件和一个 Button 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:orientation = "vertical"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent">
  <TextView
    android:id = "@ + id/myTextView2"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:textSize = "16sp"
    android:gravity = "center"
    android:padding = "10px"/>
  <Button
    android:id = "@ + id/myButton2"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "返回"/>
</LinearLayout>
```

(4) 打开 src\fs. li9_4capacity 包下的 MainActivity. java 文件,在文件中实现查看电池剩余量。代码为:

```
package fs.li9_4capacity;
import android.app.Activity;
import android.app.Dialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
```

```
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;
public class MainActivity extends Activity
{
  /* 声明变量 */
  private int intLevel;
  private int intScale;
  private Button mButton01;
  /* 创建 BroadcastReceiver */
  private BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver()
  {
    public void onReceive(Context context, Intent intent)
    {
      String action = intent.getAction();
      /* 如果捕捉到的 action 是 ACTION_BATTERY_CHANGED,
       * 就运行 onBatteryInfoReceiver() */
      if (Intent.ACTION_BATTERY_CHANGED.equals(action))
      {
        intLevel = intent.getIntExtra("level", 0);
        intScale = intent.getIntExtra("scale", 100);
        onBatteryInfoReceiver(intLevel,intScale);
      }
    }
  };
  /* 第一次调用活动 */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /* 载入 main.xml Layout */
    setContentView(R.layout.main);
    /* 初始化 Button,并设置单击后的动作 */
    mButton01 = (Button)findViewById(R.id.myButton1);
    mButton01.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View v)
      {
        /* 注册一个系统 BroadcastReceiver,作为访问电池容量之用 */
        registerReceiver
        (
          mBatInfoReceiver,
          new IntentFilter(Intent.ACTION_BATTERY_CHANGED)
        );
      }
    });
  }
  /* 捕捉到 ACTION_BATTERY_CHANGED 时要运行的 method */
  public void onBatteryInfoReceiver(int intLevel, int intScale)
  {
    /* create 跳出的对话框 */
    final Dialog d = new Dialog(MainActivity.this);
    d.setTitle(R.string.str_dialog_title);
    d.setContentView(R.layout.exadialog);
    /* 创建一个背景模糊的 Window,且将对话框放在前面 */
```

```
        Window window = d.getWindow();
        window.setFlags
        (
          WindowManager.LayoutParams.FLAG_BLUR_BEHIND,
          WindowManager.LayoutParams.FLAG_BLUR_BEHIND
        );
        /* 将取得的电池容量显示于 Dialog 中 */
        TextView mTextView02 = (TextView)d.findViewById(R.id.myTextView2);
        mTextView02.setText
        (
          getResources().getText(R.string.str_dialog_body) +
          String.valueOf(intLevel * 100 / intScale) + "%"
        );
        /* 设置返回主界面的按钮 */
        Button mButton02 = (Button)d.findViewById(R.id.myButton2);
        mButton02.setOnClickListener(new Button.OnClickListener()
        {
          @Override
          public void onClick(View v)
          {
            /* 反注册 Receiver,并关闭对话框 */
            unregisterReceiver(mBatInfoReceiver);
            d.dismiss();
          }
        });
        d.show();
      }
    }
```

运行程序,效果如图 9-4(a)所示。当单击界面中的“获取”按钮时,显示剩余的电量,效果如图 9-4(b)所示。



(a) 默认界面



(b) 查看剩余电量

图 9-4　查看手机剩余电量

9.5 接收到短信的提示

Android 手机在接收到短信时，不仅仅可以提醒用户收到一条短信，还可以自定义实现阅读短信信息、将短信发送人以及短信的内容显示到自定义界面上。下面通过一个实例来实现接收到短信的提示。

【例 9-5】 本例分为两个界面，首先进入的是主界面，在主界面中等待接收短信。当手机接收到短信时，将短信的信息重新组合，获取发件人电话和短信内容。然后发送 Intent 返回 Activity，并判断 bundle 是否为空，如果不为空，则切换界面进入短信信息界面，并在该界面显示短信的详细信息。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li9_5Receiver。

(2) 打开 res\layout 目录下的 main.xml 文件，用于创建主界面，在文件中声明一个 TextView 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#bbbccc">
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "等待短信中....." />
</RelativeLayout>
```

(3) 在 res\layout 目录下创建一个 secondmain.xml 文件，用于实现第 2 个界面，在该文件中声明实现线性布局、两个 TextView 控件、两个 EditText 控件。代码为：

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <LinearLayout
        android:id = "@+id/LinearLayout01"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        android:background = "#aacc66">
        <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
```

```
            android:text = "发信人号码: "
            android:textSize = "20dip"
            android:textColor = "#FFFFFF"
            android:textStyle = "bold"/>
        <EditText
            android:id = "@+id/EditText1"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content">
        </EditText>
    </LinearLayout>
    <LinearLayout
        android:id = "@+id/LinearLayout2"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        android:background = "#ffcc66">
        <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "短信内容: "
            android:textSize = "20dip"
            android:textColor = "#FFFFFF"
            android:textStyle = "bold"/>
        <EditText
            android:id = "@+id/EditText2"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content">
        </EditText>
    </LinearLayout>
</LinearLayout>
```

(4) 打开 src\fs.li9_5receiver 包下的 MainActivity.java 文件，在文件中实现 Activity 类的开发。代码为：

```
package fs.li9_5receiver;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.EditText;
public class MainActivity extends Activity {
    /* 第一次调用 Activity 活动 */
    //创建 Handle 对象,接收信息,并完成界面间的切换
    Handler hd = new Handler()
    {
        @Override
        public void handleMessage(Message msg)
        {
            switch(msg.what)
            {
                case 0:
                    Bundle b = msg.getData();
                    String tempMsg = (String) b.get("secondmain");
```

```
                    gotoXX(tempMsg);                    //进入信息界面
                break;
            }
        }
    };
    @Override
    //获取发送来的 bundle,如果 bundle 为空则进入主界面; 如果不为空,则取出短信信息,创建
    //Bundle 对象并绑定信息,通过 Handle 发送消息
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Bundle bundle = this.getIntent().getExtras();     //取得短信发送来的 bundle
        if(bundle!= null)
        {
            String tempMsg = bundle.getString("change");  //获取信息
            Bundle b = new Bundle();
            b.putString("secondmain", tempMsg);
            Message m = new Message();
            m.what = 0;
            m.setData(b);
            hd.sendMessage(m);
        }else
        {
             setContentView(R.layout.main);               //设置主界面
        }

    }
    //短信信息界面,首先设置当前界面,然后创建 EditText 对象,获取短信信息,并将电话号码和
    //短信内容设置在该界面的 EditText 中
    public void gotoXX(String msg)
    {
        setContentView(R.layout.secondmain);              //设置当前界面
        EditText et = (EditText)this.findViewById(R.id.EditText1); //创建对象
        EditText ett = (EditText)this.findViewById(R.id.EditText2);
        String[] tempMsg = msg.split("\\|");
        et.setText(tempMsg[0]);                           //设置电话号码
        ett.setText(tempMsg[1]);                          //设置短信内容
    }
}
```

(5) 在 src\fs.li9_5receiver 包下创建一个 SecondMainActivity.java 文件,该文件用于创建并继承 BroadcastReceiver 类。在该类中对接收信息进行过滤,将收到的短信内容重新组织,通过 Intent 返回到 Activity 的开发。代码为:

```
package fs.li9_5receiver;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.gsm.SmsMessage;
import android.util.Log;
import android.widget.Toast;
@SuppressWarnings("deprecation")
public class SecondMainActivity extends BroadcastReceiver{
    @Override
```

```
    //过滤 Action 为 android.provider.Telephony.SMS_RECEIVED 事件,当条件成立时,创建
    //StringBuilder 和 Bundle 对象,并获取短信信息
    public void onReceive(Context context, Intent intent) {
        // TODO 自动生成的方法存根
        if(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED"))
        {
            StringBuilder sb = new StringBuilder();
            Bundle bundle = intent.getExtras();           //创建 Bundle 对象,获取信息
            if(bundle!= null)
            {
                Object[] obj = (Object[])bundle.get("pdus");
                SmsMessage[] sm = new SmsMessage[obj.length];
                int length = obj.length;
                Log.d("length", length + "");
                for(int i = 0;i < length;i++)
                {
                    sm[i] = SmsMessage.createFromPdu((byte[])obj[i]);
                }
                //重新组织获取的短信信息,然后创建 Toast,显示提示信息
                for(int i = 0;i < length;i++)
                {

                    sb.append(sm[i].getDisplayOriginatingAddress()); //电话号码
                    sb.append("|");
                    sb.append(sm[i].getMessageBody());    //短信信息
                }
                Toast.makeText(
                        context,
                        "接收到一条短信!",
                        Toast.LENGTH_SHORT
                        ).show();
                //创建 Intent 和 Bundle 对象,使用 Bundle 绑定信息,设置新的 task,最后将绑定
                //好的信息通过 Intent 启动 Activity
            Intent tempIntent = new Intent(context,MainActivity.class);    //创建 Intent 对象
                Bundle myBundle = new Bundle();
                myBundle.putString("change", sb.toString().trim());
                tempIntent.putExtras(myBundle);
        tempIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);//设置新的 task
                context.startActivity(tempIntent);       //启动 Actvity
            }
        }
    }
}
```

(6) 打开 AndroidManifest.xml 文件,声明 Receiver,用于聆听系统广播信息、过滤事件等,并设置接收短信的权限。代码为:

```
…
        </activity>
    <!-- 创建 Receive 聆听系统广播信息 -->
        <receiver android:name = ".MyReceiver6_5">
            <intent-filter>
                <action android:name = "android.provider.Telephony.SMS_RECEIVED"></action>
            </intent-filter>
```

```
        </receiver>
    </application>
    <!-- 设置权限 -->
    <uses-permission android:name="android.permission.RECEIVE_SMS"></uses-permission>
</manifest>
```

运行程序，效果如图 9-5 所示。如果要发送短信，可打开 Android 的 DDMS 下的 Emulator Control，实现短信的发送，效果如图 9-6 所示。



图 9-5　接收短信的主界面

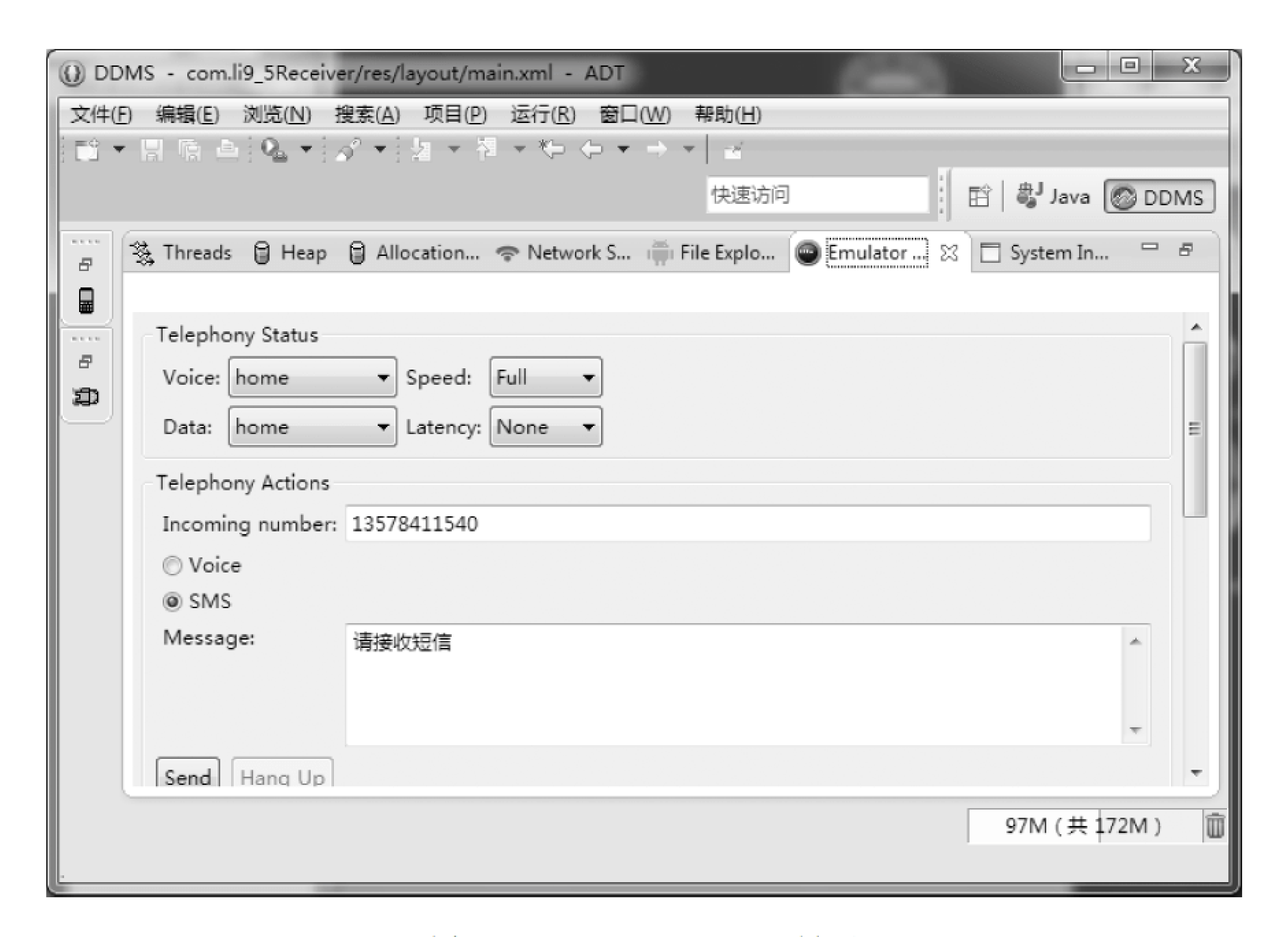


图 9-6　Emulator Control 界面

9.6　短信防火墙

大家常常会收到各种各样的垃圾短信,短信防火墙是短信软件中非常常见的功能。在 Android 系统中存在很多系统广播,当手机接收到短信时,就是通过使用广播的方式来通知所有的应用程序的。而且,短信广播是一个有序的广播,一次传递给一个广播接收器,当该接收器处理完成后才会传递给下一个接收器。这样,就可以通过在系统短信程序接收到短信广播之前终止该短信广播,从而实现短信防火墙。

【例 9-6】 实现短信防火墙。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li9_6Firewall。

(2) 对于短信防火墙这类软件是不需要任何界面的,当有短信广播时,接收到广播进行短信判断。在 src\fs.li9_6firewall 包下建立一个 SMS_rece.java 文件,代码为:

```
package fs.li9_6firewall;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.Toast;
//接收到短信
public class SMS_rece extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO 自动生成的方法存根
        String action = intent.getAction();
        //在 Android 中提供了 SmsMessage 类来管理获取的信息,从短信广播中获取 pdu 数据并转
        //换为 SmsMessage 类
        if (action.equals(MainActivity.SMS_RECEIVER)) {
            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                Object[] object = (Object[]) bundle.get("pdus");
                SmsMessage[] messages = new SmsMessage[object.length];
                for (int i = 0; i < object.length; i++) {
                    messages[i] = SmsMessage.createFromPdu((byte[]) object[i]);
                }
                SmsMessage message = messages[0];
                //显示获取的短信号码以及短信内容
                Toast.makeText(
                        context,
                        "接收到消息的号码是: " + message.getDisplayOriginatingAddress()
                                + "\n 接收到的消息是" + message.getMessageBody(), 1000)
                        .show();
                Log.i(MainActivity.TAG, "接收到消息的号码是: "
                    + message.getDisplayOriginatingAddress() + ", 接收到的消息是"
                        + message.getMessageBody());
                //通过短信号码进行拦截.如果短信号码为 5566,则禁止该短信广播,这样系统短
```

```
                    //信程序将接收不到该广播,不会提示有新的短信,从而达到短信防火墙的目的
                    if (message.getDisplayOriginatingAddress().equals("5566")) {
                        abortBroadcast();
                        Log.i(MainActivity.TAG, "终止了短信广播");
                    }
                }
            }
        }
}
```

(3) 打开 src\fs.li9_6firewall 包下的 MainActivity.java,实现 Activity 类。代码为:

```
package fs.li9_6firewall;
import java.util.ArrayList;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.Uri;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
    static final String TAG = "EXSMS";
    private static final String SENT_SMS_ACTION = "SENT_SMS_ACTION";
    private static final String DELIVERED_SMS_ACTION = "DELIVERED_SMS_ACTION";
    public static final String SMS_RECEIVER = "android.provider.Telephony.SMS_RECEIVED";
    Button btn_sys_send, btn_send;
    EditText in_ph_num, in_sms_text;
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

(4) 打开 AndroidManifest.xml 文件,设置短信防火墙权限。代码为:

```
…
        </activity>
    <!-- 广播注册 -->
        <receiver android:name=".SMS_receiver">
            <intent-filter android:priority="10000">
              <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>
    </application>
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

9.7 语 音 识 别

语音识别在 Android 上使用起来很方便,也很简单。Android 中主要通过 RecognizerIntent 来实现语音识别,其实现代码比较简单,但是如果找不到语音识别设备,就会抛出异常 ActivityNotFoundException,所以需要捕捉这个异常。另外,语音识别在模拟器上是无法测试的,因为语音识别是访问 Google 云端数据,所以如果手机的网络没有开启,就无法实现语音识别,一定要开启手机的网络,如果手机不存在语音识别功能,也是无法启用语音识别的。

下面通过一个实例来演示怎样实现手机的语音识别。

【例 9-7】 实现手机语音识别。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li9_7Voice。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 Button 控件。代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#bbbccc">
    <Button
        android:id = "@ + id/btn"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentTop = "true"
        android:layout_marginLeft = "15dp"
        android:layout_marginTop = "18dp"
        android:text = "开启语音识别" />
</RelativeLayout>
```

(3) 打开 src\fs.li9_7voice 包下的 MainActivity.java 文件,在文件中实现语音识别功能。代码为:

```
package fs.li9_7voice;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.view.View;
import android.view.View.OnClickListener;
```

```java
import android.widget.Button;
import android.widget.Toast;
import java.util.ArrayList;
import java.util.List;
public class MainActivity extends Activity implements OnClickListener {
    private static final int VOICE_RECOGNITION_REQUEST_CODE = 1234;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button) findViewById(R.id.btn);    // 识别按钮
        PackageManager pm = getPackageManager();
        List activities = pm.queryIntentActivities(new Intent(
            RecognizerIntent.ACTION_RECOGNIZE_SPEECH),0);// 本地识别程序
        // new Intent(RecognizerIntent.ACTION_WEB_SEARCH), 0);  //网络识别程序
          /*
         * 此处没有使用捕捉异常,而是检测是否有语音识别程序
       * 也可以在 startRecognizerActivity()方法中捕捉 ActivityNotFoundException 异常
          */
        if (activities.size()!= 0) {
            btn.setOnClickListener(this);
        } else {
            // 若检测不到语音识别程序在本机安装,则将按钮置灰
            btn.setEnabled(false);
            btn.setText("未检测到语音识别设备");
        }
    }
    public void onClick(View v) {
        if (v.getId() == R.id.btn) {
            startRecognizerActivity();
        }
    }
    // 开始识别
    private void startRecognizerActivity() {
        // 通过 Intent 传递语音识别的模式,开启语音
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        // 语言模式和自由模式的语音识别
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
        // 提示语音开始
        intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "开始语音");
        // 开始语音识别
        startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
        // 调出识别界面
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        // 回调获取从 Google 得到的数据
        if (requestCode == VOICE_RECOGNITION_REQUEST_CODE
                && resultCode == RESULT_OK) {
            // 取得语音的字符
            ArrayList<String> results = data
                    .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            String resultString = "";
```

```
                for (int i = 0; i < results.size(); i++) {
                    resultString += results.get(i);
                }
                Toast.makeText(this, resultString, Toast.LENGTH_SHORT).show();
            }
            //语音识别后的回调,将识别的字串以 Toast 显示
            super.onActivityResult(requestCode, resultCode, data);
        }
}
```

(4) 打开 AndroidManifest.xml 文件,设置语音权限。代码为:

```
…
    </application>
    <!-- 添加权限 -->
<uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

9.8 计算器的实现

计算器是当今所有手机上都集成拥有的功能,Android 手机也不例外。Android 手机的计算器功能越来越多,使用越来越方便。下面通过一个实例来演示计算器界面的生成。

【例 9-8】 实现计算器界面。

其实现步骤如下:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li9_8Calculators。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中实现相应的控件声明和计算器界面。代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#aaaccc">
    <LinearLayout android:layout_width="fill_parent"
    android:layout_height="wrap_content">
<TextView
     android:id="@+id/tvResult"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:height="50dp"
     android:text="" />
</LinearLayout>
<LinearLayout android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <Button
        android:id="@+id/btnBackspace"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="150dp"
```

```
            android:layout_marginLeft = "10dp"
                 android:text = "backspace"/>
          < Button
            android:id = "@ + id/btnCE"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
             android:width = "150dp"
                 android:text = "CE"/>
</LinearLayout >
< LinearLayout android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
          < Button
            android:id = "@ + id/btn7"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "10dp"
             android:width = "75dp"
             android:text = "7"/>
           < Button
            android:id = "@ + id/btn8"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = "8"/>
          < Button
            android:id = "@ + id/btn9"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = "9"/>
          < Button
            android:id = "@ + id/btnDiv"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = "/"/>
</LinearLayout >
< LinearLayout android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
          < Button
            android:id = "@ + id/btn4"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "10dp"
            android:width = "75dp"
            android:text = "4"/>
          < Button
            android:id = "@ + id/btn5"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = "5"/>
          < Button
            android:id = "@ + id/btn6"
```

```
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = "6"/>
      < Button
        android:id = "@ + id/btnMul"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = " * 1"/>
</LinearLayout >
< LinearLayout android:layout_width = "fill_parent"
    android:layout_height = "wrap_content">
      < Button
        android:id = "@ + id/btn1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginLeft = "10dp"
        android:width = "75dp"
        android:text = "1"/>
     < Button
        android:id = "@ + id/btn2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = "2"/>
     < Button
        android:id = "@ + id/btn3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = "3"/>
      < Button
        android:id = "@ + id/btnAdd"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = " + "/>
</LinearLayout >
< LinearLayout android:layout_width = "fill_parent"
    android:layout_height = "wrap_content">
      < Button
        android:id = "@ + id/btn0"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginLeft = "10dp"
        android:width = "75dp"
        android:text = "0"/>
     < Button
        android:id = "@ + id/btnC"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = "C"/>
```

```
        <Button
            android:id = "@ + id/btnEqu"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = " = "/>
        <Button
            android:id = "@ + id/btnSub"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = " - "/>
</LinearLayout>
</LinearLayout>
```

(3) 打开 src\fs. li9_8calculators 包下的 MainActivity. java 文件，在文件中实现数据的相应运算。代码为：

```
package fs. li9_8calculators;
import android. os. Bundle;
import android. view. View;
import android. view. View. OnClickListener;
import android. widget. Button;
import android. widget. TextView;
import android. app. Activity;
public class MainActivity extends Activity implements OnClickListener{
    //声明一些控件
    Button btn0 = null;
    Button btn1 = null;
    Button btn2 = null;
    Button btn3 = null;
    Button btn4 = null;
    Button btn5 = null;
    Button btn6 = null;
    Button btn7 = null;
    Button btn8 = null;
    Button btn9 = null;
    Button btnBackspace = null;
    Button btnCE = null;
    Button btnC = null;
    Button btnAdd = null;
    Button btnSub = null;
    Button btnMul = null;
    Button btnDiv = null;
    Button btnEqu = null;
    TextView tvResult = null;
    //声明两个参数,接收 tvResult 前后的值
    double num1 = 0,num2 = 0;
    double Result = 0;                          //计算结果
    int op = 0;                                 //判断操作数
    boolean isClickEqu = false;                 //判断是否按了" = "按钮
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
        //从布局文件中获取控件
        btn0 = (Button)findViewById(R.id.btn0);
        btn1 = (Button)findViewById(R.id.btn1);
        btn2 = (Button)findViewById(R.id.btn2);
        btn3 = (Button)findViewById(R.id.btn3);
        btn4 = (Button)findViewById(R.id.btn4);
        btn5 = (Button)findViewById(R.id.btn5);
        btn6 = (Button)findViewById(R.id.btn6);
        btn7 = (Button)findViewById(R.id.btn7);
        btn8 = (Button)findViewById(R.id.btn8);
        btn9 = (Button)findViewById(R.id.btn9);
        btnBackspace = (Button)findViewById(R.id.btnBackspace);
        btnCE = (Button)findViewById(R.id.btnCE);
        btnC = (Button)findViewById(R.id.btnC);
        btnEqu = (Button)findViewById(R.id.btnEqu);
        btnAdd = (Button)findViewById(R.id.btnAdd);
        btnSub = (Button)findViewById(R.id.btnSub);
        btnMul = (Button)findViewById(R.id.btnMul);
        btnDiv = (Button)findViewById(R.id.btnDiv);
        tvResult = (TextView)findViewById(R.id.tvResult);
        //添加监听
        btnBackspace.setOnClickListener(this);
        btnCE.setOnClickListener(this);
        btn0.setOnClickListener(this);
        btn1.setOnClickListener(this);
        btn2.setOnClickListener(this);
        btn3.setOnClickListener(this);
        btn4.setOnClickListener(this);
        btn5.setOnClickListener(this);
        btn6.setOnClickListener(this);
        btn7.setOnClickListener(this);
        btn8.setOnClickListener(this);
        btn9.setOnClickListener(this);
        btnAdd.setOnClickListener(this);
        btnSub.setOnClickListener(this);
        btnMul.setOnClickListener(this);
        btnDiv.setOnClickListener(this);
        btnEqu.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        //btnBackspace 和 CE
        case R.id.btnBackspace:
            String myStr = tvResult.getText().toString();
            try {
                tvResult.setText(myStr.substring(0, myStr.length() - 1));
            } catch (Exception e) {
                tvResult.setText("");
            }
            break;
        case R.id.btnCE:
            tvResult.setText(null);
```

```
        break;
        //btn0～btn9
    case R.id.btn0:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString = tvResult.getText().toString();
        myString += "0";
        tvResult.setText(myString);
        break;
    case R.id.btn1:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString1 = tvResult.getText().toString();
        myString1 += "1";
        tvResult.setText(myString1);
        break;
    case R.id.btn2:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString2 = tvResult.getText().toString();
        myString2 += "2";
        tvResult.setText(myString2);
        break;
    case R.id.btn3:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString3 = tvResult.getText().toString();
        myString3 += "3";
        tvResult.setText(myString3);
        break;
    case R.id.btn4:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString4 = tvResult.getText().toString();
        myString4 += "4";
        tvResult.setText(myString4);
        break;
    case R.id.btn5:
        if(isClickEqu)
```

```
            {
                tvResult.setText(null);
                isClickEqu = false;
            }
            String myString5 = tvResult.getText().toString();
            myString5 += "5";
            tvResult.setText(myString5);
            break;
        case R.id.btn6:
            if(isClickEqu)
            {
                tvResult.setText(null);
                isClickEqu = false;
            }
            String myString6 = tvResult.getText().toString();
            myString6 += "6";
            tvResult.setText(myString6);
            break;
        case R.id.btn7:
            if(isClickEqu)
            {
                tvResult.setText(null);
                isClickEqu = false;
            }
            String myString7 = tvResult.getText().toString();
            myString7 += "7";
            tvResult.setText(myString7);
            break;
        case R.id.btn8:
            if(isClickEqu)
            {
                tvResult.setText(null);
                isClickEqu = false;
            }
            String myString8 = tvResult.getText().toString();
            myString8 += "8";
            tvResult.setText(myString8);
            break;
        case R.id.btn9:
            if(isClickEqu)
            {
                tvResult.setText(null);
                isClickEqu = false;
            }
            String myString9 = tvResult.getText().toString();
            myString9 += "9";
            tvResult.setText(myString9);
            break;
            //+、-、*、/ = 按钮
        case R.id.btnAdd:
            String myStringAdd = tvResult.getText().toString();
            if(myStringAdd.equals(null))
            {
                return;
```

```
        }
        num1 = Double.valueOf(myStringAdd);
        tvResult.setText(null);
        op = 1;
        isClickEqu = false;
        break;
    case R.id.btnSub:
        String myStringSub = tvResult.getText().toString();
        if(myStringSub.equals(null))
        {
            return;
        }
        num1 = Double.valueOf(myStringSub);
        tvResult.setText(null);
        op = 2;
        isClickEqu = false;
        break;
    case R.id.btnMul:
        String myStringMul = tvResult.getText().toString();
        if(myStringMul.equals(null))
        {
            return;
        }
        num1 = Double.valueOf(myStringMul);
        tvResult.setText(null);
        op = 3;
        isClickEqu = false;
        break;
    case R.id.btnDiv:
        String myStringDiv = tvResult.getText().toString();
        if(myStringDiv.equals(null))
        {
            return;
        }
        num1 = Double.valueOf(myStringDiv);
        tvResult.setText(null);
        op = 4;
        isClickEqu = false;
        break;
    case R.id.btnEqu:
        String myStringEqu = tvResult.getText().toString();
        if(myStringEqu.equals(null))
        {
            return;
        }
        num2 = Double.valueOf(myStringEqu);
        tvResult.setText(null);
        switch (op) {
        case 0:
            Result = num2;
            break;
```

```
                case 1:
                    Result = num1 + num2;
                    break;
                case 2:
                    Result = num1 - num2;
                    break;
                case 3:
                    Result = num1 * num2;
                    break;
                case 4:
                    Result = num1/num2;
                    break;
                default:
                    Result = 0;
                    break;
                }
                tvResult.setText(String.valueOf(Result));
                isClickEqu = true;
                break;
            default:
                break;
            }
        }
}
```

运行程序,效果如图 9-7 所示。



图 9-7　计算器

9.9 备忘录的实现

现在手机的一个很重要的功能就是能够实现备忘录，并且在设定的时间提醒用户。Android 平台下的手机同样可以设定备忘录，并且用户可以自己制作备忘录，以在设定好的时间提醒自己。

下面通过一个实例来实现备忘录的制作。首先需要获取的是 AlarmManager，AlarmManager 是通过 getSystemService(ALARM_SERVICE)来获取的，并且使用 set()方法设定闹钟。

【例 9-9】 实现备忘录的设置与提醒。

当运行软件时，在主界面中可以设定备忘录内容，单击按钮设置提醒的时间。每次设置一个备忘录时，都会在按钮下方记录所设定的备忘录，方便用户查看。当设定的备忘录的时间到达时，在主界面上会自动弹出一个对话框进行提示。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li9_9Memo。

(2) 打开 res\layout 目录下的 main.xml 文件，用于实现主界面。代码为：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#aaaccc">
    <LinearLayout
        android:id="@+id/LinearLayout1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_width="100dip"
            android:layout_height="wrap_content"
            android:text="备忘录内容: "/>
        <EditText
            android:text=""
            android:id="@+id/EditText1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"/>
    </LinearLayout>
    <TextView
        android:text=""
        android:id="@+id/TextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
    <Button
        android:id="@+id/Button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/LinearLayout1"
        android:layout_below="@+id/LinearLayout1"
```

```
            android:layout_marginTop = "46dp"
            android:text = "设置闹钟" />
</LinearLayout>
```


(3) 在 res\layout 目录下创建一个名为 dialog.xml 的文件,用于实现闹钟设置界面。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "220dip"
    android:layout_height = "fill_parent"
    android:background = "#FFFFFF"
    android:paddingLeft = "10dip"
    android:paddingRight = "10dip"
    android:paddingTop = "10dip"
    android:paddingBottom = "10dip"
    android:gravity = "center">
    <TextView
    android:text = "备忘录时间到了,请注意!"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:textSize = "20dip"
    android:textColor = "#FFFFFF"
    android:gravity = "left"/>
    <Button
     android:text = "关闭"
     android:id = "@ + id/mywktzOk"
     android:layout_width = "60dip"
     android:layout_height = "40dip"
     android:textSize = "18dip"
     android:gravity = "center"/>
</LinearLayout>
```

(4) 打开 src\fs.li9_9memo 包下的 MainActivity.java 文件,在该文件中继承 Activity 类的开发,在该类中主要完成的是备忘录的设置。代码为:

```
public class MainActivity extends Activity {
    EditText et;                                        //备忘录编辑框
    Button button;                                      //设置按钮
    String msg;                                         //备忘录信息
    Dialog dialog;                                      //对话框
    private final int DIALOG = 0;
    TextView tv;                                        //记录备忘录
    StringBuilder sb;
    int count;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        et = (EditText)this.findViewById(R.id.EditText1);
        button = (Button)this.findViewById(R.id.Button1);
        tv = (TextView)this.findViewById(R.id.TextView1);
        Bundle bundle = this.getIntent().getExtras();    //取得短信发来的 bundle
```

```
        sb = new StringBuilder();
        if(bundle!= null)
        {
            showDialog(DIALOG);                           //显示对话框
        }
        final Calendar c = Calendar.getInstance();
        button.setOnClickListener                         //设置按钮监听器
        (
            new OnClickListener()
            {
                public void onClick(View v) {
                    msg = et.getText().toString().trim(); //获取备忘录信息
                    sb.append(count++);
                    sb.append(".备忘录内容为:");
                    sb.append(msg);
                    sb.append("\n");
                    tv.setText(sb.toString().trim());
        c.setTimeInMillis(System.currentTimeMillis());    //将当前事件设置为默认时间
                    int hour = c.get(Calendar.HOUR_OF_DAY);//小时
                    int minute = c.get(Calendar.MINUTE);  //分钟
                    new TimePickerDialog(
                            MainActivity.this,
                            new TimePickerDialog.OnTimeSetListener() {
                public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                                // TODO 自动生成的方法存根
            c.setTimeInMillis(System.currentTimeMillis());//设置当前时间
                        c.set(Calendar.HOUR_OF_DAY, hourOfDay);//设置小时
            c.set(Calendar.MINUTE, minute);               //设置分钟
            c.set(Calendar.SECOND, 0);                    //设置秒
            c.set(Calendar.MILLISECOND, 0);               //设置毫秒
            Intent intent = new Intent(MainActivity.this,AlarmRece.class);
                                                          //运行 AlarmReceiver 类
            PendingIntent pi = PendingIntent.getBroadcast( //创建 PendingIntent 对象
                MainActivity.this, 0, intent, 0); AlarmManager alarm = (AlarmManager)
MainActivity.this.getSystemService(ALARM_SERVICE);
            alarm.set(AlarmManager.RTC_WAKEUP, c.getTimeInMillis(), pi);
                                                          //设置闹钟提醒一次
alarm.setRepeating(AlarmManager.RTC_WAKEUP, c.getTimeInMillis(), 120000, pi);
                                                          //每两分钟提醒一次
String tempHour = (hourOfDay + "").length()> 1?hourOfDay + "":"0" + hourOfDay;
String tempMinute = (minute + "").length()> 1?minute + "":"0" + minute;
                Toast.makeText(MainActivity.this, "设置的时间为:" + tempHour + ":" +
tempMinute,
              Toast.LENGTH_SHORT).show();
                        }
                    },hour,minute,true).show();
                }
            }
        );
    }
    @Override
    public Dialog onCreateDialog(int id)
    {
        Dialog result = null;
```

```
            switch(id)
            {
            case DIALOG:                                    //初始化
                    AlertDialog.Builder mywktzb = new AlertDialog.Builder(this);
              mywktzb.setItems(
                    null,
                    null );
                dialog = mywktzb.create();
                result = dialog;
              break;
            }
            return result;
        }
        @Override
        public void onPrepareDialog(int id, final Dialog dialog)
        {
            switch(id)
            {
            case DIALOG:                                    //对话框
                dialog.setContentView(R.layout.dialog);
                Button mywktzBOk = (Button)dialog.findViewById(R.id.mywktzOk);
                mywktzBOk.setOnClickListener
                (
                    new OnClickListener()
                    {
                        public void onClick(View arg0) {
                            dialog.cancel();
                            MainActivity.this.finish();
                        }
                    }
                );
              dialog.setCancelable(true);               //设置在 dialog 界面中可以单击返回键
             break;
            }
        }
    }
```

(5) 在 src\fs.li9_9memo 包下创建一个 AlarmRece.java 文件，在文件中主要实现 Intent 对象的创建及启动 Activity 活动。代码为：

```
    package fs.li9_9memo;
    import android.content.BroadcastReceiver;
    import android.content.Context;
    import android.content.Intent;
    import android.os.Bundle;
    public class AlarmRece extends BroadcastReceiver{
        @Override
        public void onReceive(Context context, Intent intent) {
            Intent tempIntent = new Intent(context,MainActivity.class);        //创建 Intent 对象
            Bundle myBundle = new Bundle();
            myBundle.putString("msg", "msg");
            tempIntent.putExtras(myBundle);
            tempIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);//设置新的 task
            context.startActivity(tempIntent);                 //启动 Activity
        }
    }
```

运行程序,效果如图 9-8(a)所示,在备忘录内容右侧的文本框中填写对应的备忘内容,并单击主界面中的"设置闹钟"按钮,即可弹出闹钟设置界面,如图 9-8(b)所示,设置完成时间后,效果如图 9-8(c)所示。



(a) 备忘录主界面　　(b) 闹钟设置界面　　(c) 完成备忘录设置界面

图 9-8　设置备忘录

第 10 章　Android 的多媒体功能

Android 中的多媒体框架基于第三方公司 PackerVideo 的 OpenCore 实现，支持所有通用的音频、视频、静态图像格式。Android 平台的音频/视频采集、播放的操作都是通过 OpenCore 来实现的，OpenCore 是 Android 多媒体框架的核心。OpenCore 多媒体框架有一套通用、可扩展的接口针对第三方的多媒体编/解码、输入、输出设备等，支持多媒体文件的播放、下载。

10.1　音频播放

在 Android 系统中使用的底层框架库提供了对大部分图像和音频/视频编码格式的支持，主要包括 MPEG4、H.264、MP3、AAC、AMR、JPG、PNG、GIF 等格式。当然，要完全支持这些格式还需要硬件设备的支持。

在多媒体播放中，Android 系统使用了一个名为 MediaPlayer 的类。该类可以用来播放音频、视频和流媒体，MediaPlayer 包含了音频(Audio)和视频(Video)的播放功能。

对于播放的文件来源，可以是本地文件系统中的文件、外部存储设备文件以及网络文件。

下面通过几个实例来实现这些文件源的音频播放。

【例 10-1】 实现从资源文件中播放音乐。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li10_1Resource。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明三个 Button 控件、一个 TextView 控件。代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="#aaaccc">
    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
```

```
        android:layout_height = "wrap_content"
        android:orientation = "horizontal" >
    </LinearLayout>
    <Button
        android:id = "@ + id/button_start"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/mp3_name"
        android:layout_alignRight = "@ + id/linearLayout1"
        android:layout_below = "@ + id/mp3_name"
        android:layout_marginTop = "68dp"
        android:text = "播放" />
    <TextView
        android:id = "@ + id/mp3_name"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/linearLayout1"
        android:layout_below = "@ + id/linearLayout1"
        android:layout_marginTop = "34dp"
        android:text = "Large Text"
        android:textAppearance = "?android:attr/textAppearanceLarge" />
    <Button
        android:id = "@ + id/button_pause"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/button_start"
        android:layout_alignRight = "@ + id/button_start"
        android:layout_below = "@ + id/button_start"
        android:text = "暂停" />
    <Button
        android:id = "@ + id/button_stop"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/button_pause"
        android:layout_alignRight = "@ + id/button_pause"
        android:layout_below = "@ + id/button_pause"
        android:text = "停止" />
</RelativeLayout>
```

(3) 在 res 目录下创建一个名为 raw 的新文件夹，并将播放的音频文件 lovesong. mp3 添加到 raw 文件夹中。

(4) 打开 src\li10_1resource 包下的 MainActivity. java 文件，实现音频的播放、暂停以及停止的控制操作。代码为：

```
package fs.li10_1resource;
import java.io.File;
import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
//初始化全局变量
public class MainActivity extends Activity {
    private MediaPlayer mediaPlayer = null;              //创建一个空 MediaPlayer 对象
    private Button startButton = null;                   //播放 Button 控件对象
    private Button pauseButton = null;                   //暂停 Button 控件对象
    private Button stopButton = null;                    //停止 Button 控件对象
    private TextView nameTextView = null;                //文件名称 TextView 控件对象
    private boolean isPause = false;                     //是否暂停
    /* 第一次调用 Activity 活动 */
    @Override
    //重写 onCreate()方法,实现界面布局以及控件绑定
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
       //实例化文件名称 TextView 控件对象
        nameTextView = (TextView) findViewById(R.id.mp3_name); nameTextView.setText("
lovesong");                                              //设置文件名称
startButton = (Button) findViewById(R.id.button_start);  //实例化播放 Button 控件对象
      //添加"播放"按钮单击事件监听
        startButton.setOnClickListener(new Button.OnClickListener() {
              @Override
                    public void onClick(View arg0) {
                        start();                         //调用 MP3 播放方法
                    }
                });
pauseButton = (Button) findViewById(R.id.button_pause);  //实例化暂停 Button 控件对象
       //添加"暂停"按钮单击事件监听
        pauseButton.setOnClickListener(new Button.OnClickListener() {
        @Override
                    public void onClick(View arg0) {
                        pause();                         //调用 MP3 暂停播放方法
                    }
                });
stopButton = (Button) findViewById(R.id.button_stop);    //实例化停止 Button 控件对象
        //添加"停止"按钮单击事件监听
        stopButton.setOnClickListener(new Button.OnClickListener() {
                @Override
                    public void onClick(View arg0) {
                        stop();                          //调用 MP3 停止播放方法
                    }
                });
    }
    // MP3 开始播放方法
    public void start() {
        try {
            if (mediaPlayer != null) {                   //判断 MediaPlayer 对象不为空
      //判断 MediaPlayer 对象正在播放中,并不执行以下程序
                if (mediaPlayer.isPlaying()) {
                    return;
                }
            }
            stop();                                      //调用停止播放方法
            if (isPause) {          //判断 MediaPlayer 对象是否暂停,如果暂停就不重新播放
                return;
```

```
            }
            mediaPlayer = new MediaPlayer();
               String path = "http://zhangmenshiting2.baidu.com/data2/music/10547672/
10547672.mp3?xcode=4013468857a89a277cf2f0741d8293f2&mid=0.6233120560 8975";
            mediaPlayer.setDataSource(path);            //为 MediaPlayer 设置数据源
            mediaPlayer.prepare();                      //准备播放
            mediaPlayer.start();                        //开始播放
            //文件播放完毕监听事件
            mediaPlayer
                    .setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
                        @Override
            public void onCompletion(MediaPlayer arg0) {    //覆盖文件播出完毕事件
            //解除资源与 MediaPlayer 的赋值关系,让资源可以被其他程序利用
                            mediaPlayer.release();
                            startButton.setText("播放");
                            isPause = false;            //取消暂停状态
                            mediaPlayer = null;
                        }
                    });
            //文件播放错误监听
            mediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener() {
                @Override
                public boolean onError(MediaPlayer arg0, int arg1, int arg2) {
                //解除资源与 MediaPlayer 的赋值关系,让资源可以被其他程序利用
                    mediaPlayer.release();
                    return false;
                }
            });
            startButton.setText("正在播放");
            pauseButton.setText("暂停");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    // MP3 播放暂停方法
    public void pause() {
        try {
            if (mediaPlayer != null) {                  //判断 MediaPlayer 对象不为空
                if (mediaPlayer.isPlaying()) {          //判断 MediaPlayer 对象正在播放中
                    mediaPlayer.pause();                //暂停播放
                    pauseButton.setText("取消暂停");
                    isPause = true;                     //暂停状态
                } else {
                    mediaPlayer.start();                //开始播放
                    pauseButton.setText("暂停");
                    isPause = false;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    // MP3 停止播放方法
    public void stop() {
```

```
            try {
                if (mediaPlayer != null) {                    //判断 MediaPlayer 对象不为空
                    mediaPlayer.stop();                       //停止播放
                    startButton.setText("播放");
                    pauseButton.setText("暂停");
                    isPause = false;                          //取消暂停状态
                    mediaPlayer.release();
                    mediaPlayer = null;
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

对于以上代码，需要注意下面两点。

(1) MediaPlayer 声明周期：对于一个 MediaPlayer 类对象，需要设置数据来源、准备数据才能进行播放。在播放过程中，可以控制其处于播放、暂停、停止等状态；在不需要播放时，可以播放该 MediaPlayer 类对象。

(2) 播放监听事件：在音频播放过程中会出现各种状态，最常见的就是文件播放结束以及发送错误。在 MediaPlayer 类中通过设置 OnCompletionListener()和 OnErrorListener()事件处理播放结束与发生错误的事件处理，在播放结束或发生错误时，都必须调用 MediaPlayer.release()方法将相关文件与资源释放出来，以避免 MediaPlayer 的资源占用。

运行程序，效果如图 10-1 所示。



图 10-1　音乐播放界面

例 10-1 中实现了从资源文件中播放音频的功能。例 10-2 主要讲解 MediaPlayer 对象加载外部 MP3 媒体文件的方式。对于加载外部媒体文件，主要通过 MediaPlayer. setDataSource()方法来实现，构建 setDataSource()的方法有很多，比较简单的方法就是直接传入 MP3 媒体文件的路径。

【例 10-2】 插入 5 个按钮，分别用于上一首、停止、播放、暂停、下一首操作。

其实现步骤如下：

(1) 在 Eclipse 中建立一个 Android 应用项目，命名为 li10_2SDSource。

(2) 编写布局文件，用于实现 5 个按钮及列表框。打开 res\layout 目录下的 main. xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< AbsoluteLayout
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:orientation = "vertical"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent"
  android:background = "@drawable/h">
< ImageButton
  android:id = "@ + id/LastImageButton"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_x = "10px"
  android:layout_y = "70px"
  android:src = "@drawable/previous"/>
< ImageButton
  android:id = "@ + id/StopImageButton"
  android:layout_height = "wrap_content"
  android:layout_width = "wrap_content"
  android:layout_x = "90px"
  android:layout_y = "70px"
  android:src = "@drawable/stop"/>
< ImageButton
  android:id = "@ + id/StartImageButton"
  android:layout_height = "wrap_content"
  android:layout_width = "wrap_content"
  android:layout_x = "170px"
  android:layout_y = "70px"
  android:src = "@drawable/play"/>
< ImageButton
  android:id = "@ + id/PauseImageButton"
  android:layout_height = "wrap_content"
  android:layout_width = "wrap_content"
  android:layout_x = "250px"
  android:layout_y = "70px"
  android:src = "@drawable/pause"/>
< ImageButton
  android:id = "@ + id/NextImageButton"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_x = "330px"
  android:layout_y = "70px"
```

```
    android:src = "@drawable/next"/>
    <ListView
        android:id = "@id/android:list"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_weight = "1"
        android:layout_x = "108dp"
        android:layout_y = "4dp"
        android:drawSelectorOnTop = "false"/>
</AbsoluteLayout>
```

(3) 实现次布局文件，在 res\layout 目录下创建一个名为 musicitme. xml 的文件，代码为：

```
<TextView xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/TextView01"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content" />
```

(4) 编写 Activity 文件，实现 5 个按钮的功能。打开 src\fs. li10_2sdsource 包下的 MainActivity. java 文件，代码为：

```
package fs.li10_2sdsource;
import java.io.File;
import java.io.FilenameFilter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import android.app.ListActivity;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.ListView;
public class MainActivity extends ListActivity
{
    //几个操作按钮
    private ImageButton mFrontImageButton = null;
    private ImageButton mStopImageButton = null;
    private ImageButton mStartImageButton = null;
    private ImageButton mPauseImageButton = null;
    private ImageButton mNextImageButton = null;
    // MediaPlayer 对象
    public MediaPlayer mMediaPlayer  =  null;
    //播放列表
    private List<String> mMusicList  =  new ArrayList<String>();
    //当前播放歌曲的索引
    private int currentListItme  =  0;
    //音乐的路径
    private static final String MUSIC_PATH  =  new String("/sdcard/");
    /* 第一次调用 Activity 活动 */
```

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    musicList();
    //构建 MediaPlayer 对象
    mMediaPlayer = new MediaPlayer();
   //获取对象
    mFrontImageButton = (ImageButton) findViewById(R.id.LastImageButton);
    mStopImageButton = (ImageButton) findViewById(R.id.StopImageButton);
    mStartImageButton = (ImageButton) findViewById(R.id.StartImageButton);
    mPauseImageButton = (ImageButton) findViewById(R.id.PauseImageButton);
    mNextImageButton = (ImageButton) findViewById(R.id.NextImageButton);
    //"停止"按钮
    mStopImageButton.setOnClickListener(new ImageButton.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            //是否正在播放
            if (mMediaPlayer.isPlaying())
            {
                //重置 MediaPlayer 到初始状态
                mMediaPlayer.reset();
            }
        }
    });
    //"播放"按钮
    mStartImageButton.setOnClickListener(new ImageButton.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            playMusic(MUSIC_PATH + mMusicList.get(currentListItme));
        }
    });
    //暂停
    mPauseImageButton.setOnClickListener(new ImageButton.OnClickListener()
    {
        public void onClick(View view)
        {
            if (mMediaPlayer.isPlaying())
            {
                //暂停
                mMediaPlayer.pause();
            }
            else
            {
                //播放
                mMediaPlayer.start();
            }
        }
    });
```

```
            //下一首
            mNextImageButton.setOnClickListener(new ImageButton.OnClickListener()
            {
                @Override
                public void onClick(View arg0)
                {
                    nextMusic();
                }
            });
            //上一首
            mFrontImageButton.setOnClickListener(new ImageButton.OnClickListener()
            {
                @Override
                public void onClick(View arg0)
                {
                    FrontMusic();
                }
            });
        }
        public boolean onKeyDown(int keyCode, KeyEvent event)
        {
            if ( keyCode == KeyEvent.KEYCODE_BACK)
            {
                mMediaPlayer.stop();
                mMediaPlayer.release();
                this.finish();
                return true;
            }
            return super.onKeyDown(keyCode, event);
        }
        @Override
        protected void onListItemClick(ListView l, View v, int position, long id)
        {
            currentListItme = position;
            playMusic(MUSIC_PATH + mMusicList.get(position));
        }
        //音乐播放列表
        public void musicList()
        {
            //取得指定位置的文件设置显示到播放列表
            File home = new File(MUSIC_PATH);
            if (home.listFiles(new MusicFilter()).length > 0)
            {
                for (File file : home.listFiles(new MusicFilter()))
                {
                    mMusicList.add(file.getName());
                }
                 ArrayAdapter<String> musicList = new ArrayAdapter<String>(MainActivity.
    this,R.layout.musicitme, mMusicList);
                setListAdapter(musicList);
            }
        }
        private void playMusic(String path)
        {
```

```
        try
        {
            //重置 MediaPlayer
            mMediaPlayer.reset();
            //设置要播放的文件的路径
            mMediaPlayer.setDataSource(path);
            //准备播放
            mMediaPlayer.prepare();
            mMediaPlayer.start();
            mMediaPlayer.setOnCompletionListener(new OnCompletionListener()
            {
                public void onCompletion(MediaPlayer arg0)
                {
                    //播放完一首之后播放下一首
                    nextMusic();
                }
            });
        }catch (IOException e){}
    }
    //下一首
    private void nextMusic()
    {
        if (++currentListItme >= mMusicList.size())
        {
            currentListItme = 0;
        }
        else
        {
            playMusic(MUSIC_PATH + mMusicList.get(currentListItme));
        }
    }
    //上一首
    private void FrontMusic()
    {
        if ( -- currentListItme >= 0)
        {
            currentListItme = mMusicList.size();
        }
        else
        {
            playMusic(MUSIC_PATH + mMusicList.get(currentListItme));
        }
    }
}
//过滤文件类型
class MusicFilter implements FilenameFilter
{
    public boolean accept(File dir, String name)
    {
        //这里还可以设置其他格式的音乐文件
        return (name.endsWith(".mp3"));
    }
}
```

在程序编写完成之后,下载两个扩展名为.mp3的歌曲,然后放到SD卡中进行测试,就可以看到效果了。

随着3G技术的逐渐成熟,移动互联网时代已经到来,且网络资费不断降低,使得直接利用网络资源不再是问题,下面通过例10-3实现从网络中播放媒体文件。如果要通过网络播放媒体文件,比较简单的方法就是使用MediaPlayer.setDataSource()方法直接传入网络媒体资源文件的地址来实现。

【例10-3】 从网络中播放音频。

其实现步骤如下:

(1) 在Eclipse中创建一个Android应用项目,命名为li10_3networkfile。

(2) 打开res\layout目录下的main.xml文件,实现网络播放音频的主界面。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:layout_height = "fill_parent"
        android:layout_width = "fill_parent"
        android:background = "#aaaccc">
    < LinearLayout android:layout_height = "wrap_content"
                   android:layout_width = "fill_parent"
                   android:orientation = "vertical"
                   android:layout_gravity = "top">
     < LinearLayout android:orientation = "horizontal"
                    android:layout_gravity = "center_horizontal"
                    android:layout_marginTop = "4.0dip"
                    android:layout_height = "wrap_content"
                    android:layout_width = "wrap_content">
    < Button android:layout_width = "wrap_content"
                   android:layout_height = "wrap_content"
                   android:id = "@ + id/btnPlayUrl"
                   android:text = "播放网络音频"/>
     < Button android:layout_height = "wrap_content"
          android:id = "@ + id/btnPause"
          android:text = "暂停"
          android:layout_width = "80dip"/>
      < Button android:layout_height = "wrap_content"
                 android:layout_width = "80dip"
                 android:text = "停止"
                 android:id = "@ + id/btnStop"/>
        </LinearLayout >
        < LinearLayout android:orientation = "horizontal"
                       android:layout_width = "fill_parent"
                       android:layout_height = "wrap_content"
                       android:layout_marginBottom = "20dip">
         < SeekBar android:paddingRight = "10dip"
                       android:layout_gravity = "center_vertical"
                       android:paddingLeft = "10dip"
                       android:layout_weight = "1.0"
                       android:layout_height = "wrap_content"
                       android:layout_width = "wrap_content"
                       android:id = "@ + id/skbProgress"
                       android:max = "100"/>
        </LinearLayout >
```

```
    </LinearLayout>
</FrameLayout>
```

（3）打开 src\fs.li10_3networkfile 包下的 MainActivity.java 文件，该文件主要负责调用 Player 类，其中关键部分是 SeekBarChangeEvent 这个 SeekBar 拖动的事件。SeekBar 的 Progress 是 0～SeekBar.getMax()之内的数，而 MediaPlayer.seekTo()的参数是 0～MediaPlayer.getDuration()之内的数，所以，MediaPlayer.seekTo()的参数是(progress/seekBar.getMax()) * player.mediaPlayer.getDuration()。代码为：

```
package fs.li10_3networkfile;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.SeekBar;
public class MainActivity extends Activity {
    private Button btnPause, btnPlayUrl, btnStop;
    private SeekBar skbProgress;
    private Player player;
    /* 第一次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnPlayUrl = (Button) this.findViewById(R.id.btnPlayUrl);
        btnPlayUrl.setOnClickListener(new ClickEvent());
        btnPause = (Button) this.findViewById(R.id.btnPause);
        btnPause.setOnClickListener(new ClickEvent());
        btnStop = (Button) this.findViewById(R.id.btnStop);
        btnStop.setOnClickListener(new ClickEvent());
        skbProgress = (SeekBar) this.findViewById(R.id.skbProgress);
        skbProgress.setOnSeekBarChangeListener(new SeekBarChangeEvent());
        player = new Player(skbProgress);
    }
    class ClickEvent implements OnClickListener {
        @Override
        public void onClick(View arg0) {
            if (arg0 == btnPause) {
                player.pause();
            } else if (arg0 == btnPlayUrl) {
                //在百度 MP3 里随便搜索到的,大家可以试试别的链接
                String url = "http://219.138.125.22/myweb/mp3/CMP3/JH19.MP3";
                player.playUrl(url);
            } else if (arg0 == btnStop) {
                player.stop();
            }
        }
    }
    class SeekBarChangeEvent implements SeekBar.OnSeekBarChangeListener {
        int progress;
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress,
```

```
                    boolean fromUser) {
                //原本是(progress/seekBar.getMax()) * player.mediaPlayer.getDuration()
                this.progress = progress * player.mediaPlayer.getDuration()
                        / seekBar.getMax();
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                //seekTo()的参数是相对于影片时间的数字,而不是与 seekBar.getMax()相对的数字
                player.mediaPlayer.seekTo(progress);
            }
        }
    }
```

(4) 在 src\fs.li10_3networkfile 包下创建一个 Player.java 文件,用于实现"进度条更新"、"数据缓冲"等功能,它们虽然不是很复杂的功能,但却是非常有用的功能。代码为:

```
package fs.li10_3networkfile;
import java.io.IOException;
import java.util.Timer;
import java.util.TimerTask;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnBufferingUpdateListener;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.widget.SeekBar;
public class Player implements OnBufferingUpdateListener,
        OnCompletionListener, MediaPlayer.OnPreparedListener{
    public MediaPlayer mediaPlayer;
    private SeekBar skbProgress;
    private Timer mTimer = new Timer();
    public Player(SeekBar skbProgress)
    {
        this.skbProgress = skbProgress;
        try {
            mediaPlayer = new MediaPlayer();
            mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
            mediaPlayer.setOnBufferingUpdateListener(this);
            mediaPlayer.setOnPreparedListener(this);
        } catch (Exception e) {
            Log.e("mediaPlayer", "error", e);
        }
        mTimer.schedule(mTimerTask, 0, 1000);
    }
    //通过定时器和 Handler 来更新进度条
    TimerTask mTimerTask = new TimerTask() {
        @Override
        public void run() {
            if(mediaPlayer == null)
```

```
                return;
            if (mediaPlayer.isPlaying() && skbProgress.isPressed() == false) {
                handleProgress.sendEmptyMessage(0);
            }
        }
    };
    Handler handleProgress = new Handler() {
        public void handleMessage(Message msg) {
            int position = mediaPlayer.getCurrentPosition();
            int duration = mediaPlayer.getDuration();
            if (duration > 0) {
                long pos = skbProgress.getMax() * position / duration;
                skbProgress.setProgress((int) pos);
            }
        };
    };
    public void play()
    {
        mediaPlayer.start();
    }
    public void playUrl(String videoUrl)
    {
        try {
            mediaPlayer.reset();
            mediaPlayer.setDataSource(videoUrl);
            mediaPlayer.prepare();                    //prepare 之后自动播放
            //mediaPlayer.start();
        } catch (IllegalArgumentException e) {
            // TODO  自动生成的 catch 块
            e.printStackTrace();
        } catch (IllegalStateException e) {
            // TODO  自动生成的 catch 块
            e.printStackTrace();
        } catch (IOException e) {
            // TODO  自动生成的 catch 块
            e.printStackTrace();
        }
    }
    public void pause()
    {
        mediaPlayer.pause();
    }
    public void stop()
    {
        if (mediaPlayer != null) {
            mediaPlayer.stop();
            mediaPlayer.release();
            mediaPlayer = null;
        }
    }
    @Override
    //通过 onPrepared 播放
    public void onPrepared(MediaPlayer arg0) {
        arg0.start();
```

```
            Log.e("mediaPlayer", "onPrepared");
        }
        @Override
        public void onCompletion(MediaPlayer arg0) {
            Log.e("mediaPlayer", "onCompletion");
        }
        @Override
        public void onBufferingUpdate(MediaPlayer arg0, int bufferingProgress) {
            skbProgress.setSecondaryProgress(bufferingProgress);
             int currentProgress = skbProgress.getMax() * mediaPlayer.getCurrentPosition()/
mediaPlayer.getDuration();
            Log.e(currentProgress + "% play", bufferingProgress + "% buffer");
        }
    }
```

运行程序，效果如图 10-2 所示。



图 10-2　从网络中播放音乐的界面

10.2　录制多媒体

在 10.1 节中实现了一个 MP3 的音频播放，使用户掌握了 Android 系统中多媒体文件的播放的实现。当然，Android 提供了对多媒体的播放，自然会提供对多媒体的采样录制功能。这需要手机本身的硬件支持，Android 中的多媒体录制由 MediaRecondser 类提供了相关方法。

下面通过一个实例来演示 Android 录制多媒体。

【例 10-4】 在 Android 平台上实现录音及播放功能。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li10_4MediaRecord。

(2) 打开 res\layout 目录下的 main.xml 文件，用于实现录制声音的主界面。代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<!-- 建立一个 RadioGroup -->
<RadioGroup
    android:id="@+id/myRadioGroup3"
    android:layout_width="150px"
    android:layout_height="150px"
    android:layout_marginTop="62dp"
    android:layout_x="13dp"
    android:layout_y="-17dp"
    android:orientation="vertical" >
</RadioGroup>
<RadioButton
    android:id="@+id/myRadio2Button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_marginBottom="67dp"
    android:text="@string/radio2_op3" />
<RadioGroup
    android:id="@+id/myRadioGroup1"
    android:layout_width="150px"
    android:layout_height="150px"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/myRadioGroup3"
    android:orientation="vertical" >
    <RadioButton
        android:id="@+id/myRadio1Button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="@string/radio1_op1" />
    <RadioButton
        android:id="@+id/myRadio1Button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/radio1_op2" />
</RadioGroup>
<RadioButton
    android:id="@+id/myRadio3Button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:checked="true"
```

```
        android:text = "@string/radio3_op1" />
    < RadioButton
        android:id = "@ + id/myRadio2Button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_above = "@ + id/myRadio3Button1"
        android:layout_toLeftOf = "@ + id/btnStop2"
        android:text = "@string/radio2_op2" />
    < RadioGroup
        android:id = "@ + id/myRadioGroup2"
        android:layout_width = "150px"
        android:layout_height = "250px"
        android:layout_alignBottom = "@ + id/myRadio2Button3"
        android:layout_alignLeft = "@ + id/myRadio3Button2"
        android:orientation = "vertical" >
    </RadioGroup >
    < RadioButton
        android:id = "@ + id/myRadio3Button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignTop = "@ + id/btnTrack"
        android:text = "@string/radio3_op2" />
    < Button
        android:id = "@ + id/btnRecord"
        android:layout_width = "100px"
        android:layout_height = "100px"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentTop = "true"
        android:layout_marginTop = "14dp"
        android:text = "Record" />
    < Button
        android:id = "@ + id/btnStop"
        android:layout_width = "100px"
        android:layout_height = "100px"
        android:layout_alignBaseline = "@ + id/btnRecord"
        android:layout_alignBottom = "@ + id/btnRecord"
        android:layout_toLeftOf = "@ + id/myRadio2Button2"
        android:text = "StopR" />
    < Button
        android:id = "@ + id/btnTrack"
        android:layout_width = "100px"
        android:layout_height = "100px"
        android:layout_alignBaseline = "@ + id/btnStop"
        android:layout_alignBottom = "@ + id/btnStop"
        android:layout_alignLeft = "@ + id/myRadio2Button2"
        android:text = "Track" />
    < Button
        android:id = "@ + id/btnStop2"
        android:layout_width = "100px"
        android:layout_height = "100px"
        android:layout_alignLeft = "@ + id/myRadio2Button1"
        android:layout_alignTop = "@ + id/myRadio2Button1"
        android:text = "StopT" />
```

```
<RadioButton
    android:id = "@ + id/myRadio2Button1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignParentRight = "true"
    android:layout_below = "@ + id/myRadioGroup1"
    android:layout_marginRight = "56dp"
    android:layout_marginTop = "14dp"
    android:checked = "true"
    android:text = "@string/radio2_op1" />
</RelativeLayout>
```

(3) 打开 src\fs. li10_4mediarecord 包下的 MainActivity. java 文件，用于实现声音的录制与播放。代码为：

```
package fs. li10_4mediarecord;
import java. io. File;
import java. io. FileInputStream;
import java. io. FileNotFoundException;
import java. io. FileOutputStream;
import java. io. IOException;
import android. app. Activity;
import android. media. AudioFormat;
import android. media. AudioManager;
import android. media. AudioRecord;
import android. media. AudioTrack;
import android. media. MediaRecorder;
import android. os. Bundle;
import android. os. Environment;
import android. util. Log;
import android. view. View;
import android. widget. Button;
import android. widget. RadioButton;
import android. widget. RadioGroup;
import android. widget. SeekBar;
public class MainActivity extends Activity {
        private String TAG  =  "session";
        private RadioGroup mRadioGroup1, mRadioGroup2, mRadioGroup3;
private RadioButton mRadio1, mRadio2, mRadio3, mRadio4, mRadio5, mRadio6, mRadio7; private static
final int RECORDER_BPP  =  16;
        private static final String AUDIO_RECORDER_FOLDER  =  "AudioRecorder";
      private static final String AUDIO_RECORDER_TEMP_FILE  =  "record_temp. raw";
        private static int frequency  =  44100;
        private static int channelConfiguration  =  AudioFormat. CHANNEL_IN_STEREO;
        private static int EncodingBitRate  =  AudioFormat. ENCODING_PCM_16BIT;
        private AudioRecord audioRecord  =  null;
        private AudioTrack audioTrack  =  null;
        private int recBufSize  =  0;
        private int playBufSize  =  0;
        private Thread recordingThread  =  null;
        private boolean isRecording  =  false;
        private boolean isTracking  =  false;
        private boolean m_keep_running;
        protected PCMAudioTrack m_player;
```

```
SeekBar skbVolume;                                    //调节音量
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    setButtonHandlers();
    enableButton1(false);
}
private void setButtonHandlers() {
        ((Button)findViewById(R.id.btnRecord)).setOnClickListener(btnClick);
        ((Button)findViewById(R.id.btnStop)).setOnClickListener(btnClick);
        ((Button)findViewById(R.id.btnTrack)).setOnClickListener(btnClick);
        ((Button)findViewById(R.id.btnStop2)).setOnClickListener(btnClick);
        mRadioGroup1 = (RadioGroup) findViewById(R.id.myRadioGroup1);
        mRadioGroup2 = (RadioGroup) findViewById(R.id.myRadioGroup2);
        mRadioGroup3 = (RadioGroup) findViewById(R.id.myRadioGroup3);
        mRadio1 = (RadioButton) findViewById(R.id.myRadio1Button1);
        mRadio2 = (RadioButton) findViewById(R.id.myRadio1Button2);
        mRadio3 = (RadioButton) findViewById(R.id.myRadio2Button1);
        mRadio4 = (RadioButton) findViewById(R.id.myRadio2Button2);
        mRadio5 = (RadioButton) findViewById(R.id.myRadio2Button3);
        mRadio6 = (RadioButton) findViewById(R.id.myRadio3Button1);
        mRadio7 = (RadioButton) findViewById(R.id.myRadio3Button2);
        mRadioGroup1.setOnCheckedChangeListener(mChangeRadio1);
        mRadioGroup2.setOnCheckedChangeListener(mChangeRadio2);
        mRadioGroup3.setOnCheckedChangeListener(mChangeRadio3);
}
private void enableButton(int id,boolean isEnable){
        ((Button)findViewById(id)).setEnabled(isEnable);
}
private void enableButton0(boolean isRecording) {
    enableButton(R.id.btnRecord,!isRecording);
    enableButton(R.id.btnTrack,!isRecording);
    enableButton(R.id.btnStop2,!isRecording);
    enableButton(R.id.btnStop,isRecording);
}
private void enableButton1(boolean isRecording) {
        enableButton(R.id.btnRecord,!isRecording);
        enableButton(R.id.btnTrack,isRecording);
        enableButton(R.id.btnStop2,isRecording);
        enableButton(R.id.btnStop,isRecording);
}
private void enableButton2(boolean isRecording) {
    enableButton(R.id.btnRecord,!isRecording);
    enableButton(R.id.btnTrack,!isRecording);
    enableButton(R.id.btnStop2,isRecording);
    enableButton(R.id.btnStop,isRecording);
}
private void enableButton3(boolean isTracking) {
    enableButton(R.id.btnRecord,!isTracking);
    enableButton(R.id.btnStop,!isTracking);
    enableButton(R.id.btnTrack,!isTracking);
    enableButton(R.id.btnStop2,isTracking);
}
```

```
private void enableButton4(boolean isTracking) {
    enableButton(R.id.btnRecord,!isTracking);
    enableButton(R.id.btnStop,isTracking);
    enableButton(R.id.btnTrack,!isTracking);
    enableButton(R.id.btnStop2,isTracking);
}
private String getFilename(){
String filepath = Environment.getExternalStorageDirectory().getAbsolutePath();
        File file = new File(filepath,AUDIO_RECORDER_FOLDER);
        if(file.exists()){
             file.delete();
        }
        return (file.getAbsolutePath() + "/session.wav" );
}
private String getTempFilename(){
        String filepath = Environment.getExternalStorageDirectory().getPath();
        File file = new File(filepath,AUDIO_RECORDER_FOLDER);
        if(!file.exists()){
                file.mkdirs();
        }
        File tempFile = new File(filepath,AUDIO_RECORDER_TEMP_FILE);
        if(tempFile.exists())
                tempFile.delete();
    return (file.getAbsolutePath() + "/" + AUDIO_RECORDER_TEMP_FILE);
}
private void startRecording(){
        createAudioRecord();
        audioRecord.startRecording();
        isRecording = true;
        recordingThread = new Thread(new Runnable() {
                @Override
                public void run() {
                        writeAudioDataToFile();
                }
        },"AudioRecorder Thread");
        recordingThread.start();
}
private void writeAudioDataToFile(){
        byte data[] = new byte[recBufSize];
        String filename = getTempFilename();
        FileOutputStream os = null;
        try {
                os = new FileOutputStream(filename);
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        }
        int read = 0;
        if(null != os){
                while(isRecording){
               read = audioRecord.read(data, 0, recBufSize);
if(AudioRecord.ERROR_INVALID_OPERATION != read){
                        try {
                                os.write(data);
                        } catch (IOException e) {
```

```
                                        e.printStackTrace();
                                }
                        }
                }
                try {
                        os.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
private void stopRecording(){
        if(null != audioRecord){
                isRecording = false;
                audioRecord.stop();
                audioRecord.release();
                audioRecord = null;
                recordingThread = null;
        }
        copyWaveFile(getTempFilename(),getFilename());
        deleteTempFile();
}
private void deleteTempFile() {
        File file = new File(getTempFilename());
        file.delete();
}
private void copyWaveFile(String inFilename,String outFilename){
        FileInputStream in = null;
        FileOutputStream out = null;
        long totalAudioLen = 0;
        long totalDataLen = totalAudioLen + 36;
        long longSampleRate = frequency;
        int channels = 2;
        long byteRate = RECORDER_BPP * frequency * channels/8;
        byte[] data = new byte[recBufSize];
        try {
                in = new FileInputStream(inFilename);
                out = new FileOutputStream(outFilename);
                totalAudioLen = in.getChannel().size();
                totalDataLen = totalAudioLen + 36;
                AppLog.logString("File size: " + totalDataLen);
                WriteWaveFileHeader(out, totalAudioLen, totalDataLen,
                                longSampleRate, channels, byteRate);

                while(in.read(data) != -1){
                        out.write(data);
                }
                in.close();
                out.close();
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
```

```
}
private void WriteWaveFileHeader(
                FileOutputStream out, long totalAudioLen,
                long totalDataLen, long longSampleRate, int channels,
                long byteRate) throws IOException {
        byte[] header = new byte[44];
        header[0] = 'R'; // RIFF/WAVE header
        header[1] = 'I';
        header[2] = 'F';
        header[3] = 'F';
        header[4] = (byte) (totalDataLen & 0xff);
        header[5] = (byte) ((totalDataLen >> 8) & 0xff);
        header[6] = (byte) ((totalDataLen >> 16) & 0xff);
        header[7] = (byte) ((totalDataLen >> 24) & 0xff);
        header[8] = 'W';
        header[9] = 'A';
        header[10] = 'V';
        header[11] = 'E';
        header[12] = 'f';                       //'fmt ' chunk
        header[13] = 'm';
        header[14] = 't';
        header[15] = ' ';
        header[16] = 16;                        //4 bytes: size of 'fmt ' chunk
        header[17] = 0;
        header[18] = 0;
        header[19] = 0;
        header[20] = 1;                         //format = 1
        header[21] = 0;
        header[22] = (byte) channels;
        header[23] = 0;
        header[24] = (byte) (longSampleRate & 0xff);
        header[25] = (byte) ((longSampleRate >> 8) & 0xff);
        header[26] = (byte) ((longSampleRate >> 16) & 0xff);
        header[27] = (byte) ((longSampleRate >> 24) & 0xff);
        header[28] = (byte) (byteRate & 0xff);
        header[29] = (byte) ((byteRate >> 8) & 0xff);
        header[30] = (byte) ((byteRate >> 16) & 0xff);
        header[31] = (byte) ((byteRate >> 24) & 0xff);
        header[32] = (byte) (2 * 16 / 8);       //块对齐
        header[33] = 0;
        header[34] = RECORDER_BPP;              //每次取样位数
        header[35] = 0;
        header[36] = 'd';
        header[37] = 'a';
        header[38] = 't';
        header[39] = 'a';
        header[40] = (byte) (totalAudioLen & 0xff);
        header[41] = (byte) ((totalAudioLen >> 8) & 0xff);
        header[42] = (byte) ((totalAudioLen >> 16) & 0xff);
        header[43] = (byte) ((totalAudioLen >> 24) & 0xff);
        out.write(header, 0, 44);
}
private View.OnClickListener btnClick = new View.OnClickListener() {
        @Override
```

```
public void onClick(View v) {
        switch(v.getId()){
                case R.id.btnRecord:{
                        AppLog.logString("Start Recording");
                        enableButton0(true);
                        startRecording();
                        break;
                }
                case R.id.btnStop:{
                        AppLog.logString("Start Recording");
                        enableButton2(false);
                        stopRecording();
                        break;
                }
                case R.id.btnTrack:{
                    AppLog.logString("Start Tracking");
                    enableButton3(true);
                    m_player = new PCMAudioTrack();
                    Log.i(TAG,"++++++++++++++++++++++");
                    m_player.init();
                    m_player.start();

                    break;
                }
                case R.id.btnStop2:{
                    AppLog.logString("Stop Tracking");
                    enableButton4(false);
                    m_player.free();
                    m_player = null;
                    break;
                }
        }
    }
};
public void createAudioRecord(){
    recBufSize = AudioRecord.getMinBufferSize(frequency,
            channelConfiguration, EncodingBitRate);
audioRecord = new AudioRecord(MediaRecorder.AudioSource.MIC, frequency,
            channelConfiguration, EncodingBitRate, recBufSize);
}
public void createAudioTrack(){
    playBufSize = AudioTrack.getMinBufferSize(frequency,
            channelConfiguration, EncodingBitRate);
audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC, frequency,
            channelConfiguration, EncodingBitRate,
            playBufSize, AudioTrack.MODE_STREAM);
}
class PCMAudioTrack extends Thread {
    protected byte[] m_out_bytes;
    final String FILE_PATH = "/sdcard/AudioRecorder/";
    final String FILE_NAME = "session.wav";
    File file;
    FileInputStream in;
    public void init() {
```

```
            try {
                file = new File(FILE_PATH , FILE_NAME);
                file.createNewFile();
                in = new FileInputStream(file);
                m_keep_running = true;
                createAudioTrack();
                m_out_bytes = new byte[playBufSize];
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        public void free() {
            m_keep_running = false;
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
                Log.d("sleep exceptions...\n", "");
            }
        }
        public void run() {
            byte[] bytes_pkg = null;
            audioTrack.play();
            while (m_keep_running) {
                try {
                    in.read(m_out_bytes);
                    bytes_pkg = m_out_bytes.clone();
                    audioTrack.write(bytes_pkg, 0, bytes_pkg.length);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            audioTrack.stop();
            audioTrack = null;
            try {
                in.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    RadioGroup.OnCheckedChangeListener mChangeRadio1 =
        new RadioGroup.OnCheckedChangeListener()
      {
      @Override public void onCheckedChanged(RadioGroup group, int checkedId)
        {
          if(checkedId == mRadio1.getId())
          {
channelConfiguration = AudioFormat.CHANNEL_CONFIGURATION_STEREO;
          }
        else if(checkedId == mRadio2.getId())
        {
 channelConfiguration = AudioFormat.CHANNEL_CONFIGURATION_MONO;
        }
      }
```

```
            };
                RadioGroup.OnCheckedChangeListener mChangeRadio2 =
                    new RadioGroup.OnCheckedChangeListener()
                  {
             @Override public void onCheckedChanged(RadioGroup group, int checkedId)
                    {
                      if(checkedId == mRadio3.getId())
                      {
                          frequency = 44100;
                        }
                      else if(checkedId == mRadio4.getId())
                      {
                          frequency = 22050;
                      }
                      else if(checkedId == mRadio5.getId())
                      {
                          frequency = 11025;
                      }
                    }
                 };
                 RadioGroup.OnCheckedChangeListener mChangeRadio3 =
                        new RadioGroup.OnCheckedChangeListener()
                     {
            @Override public void onCheckedChanged(RadioGroup group, int checkedId)
                        {
                          if(checkedId == mRadio6.getId())
                          {
                            EncodingBitRate = AudioFormat.ENCODING_PCM_16BIT;
                            }
                          else if(checkedId == mRadio7.getId())
                          {
                            EncodingBitRate = AudioFormat.ENCODING_PCM_8BIT;
                          }
                        }
                    };
                    @Override
                    protected void onDestroy() {
                        super.onDestroy();
                        android.os.Process.killProcess(android.os.Process.myPid());
                    }
    }
```

(4) 在 src\fs.li10_4mediarecord 包下创建一个 AppLog.java 文件,用于返回 LogCat 输出。代码为:

```
package fs.li10_4mediarecord;
import android.util.Log;
public class AppLog {
        private static final String APP_TAG = "AudioRecorder";
        public static int logString(String message){
                return Log.i(APP_TAG,message);
        }
}
```

(5) 打开 AndroidManifest.xml 文件,用于设置权限。代码为:

```
…
    </application>
<!-- 录制与播放声音权限设置 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
</manifest>
```

运行程序,效果如图 10-3 所示。



图 10-3 录制与播放声音界面

10.3 视频播放

在 Android 系统中内置了 VideoView Widget 作为多媒体视频播放器,这样可以浏览电影视频。VideoView 类用于 VideoView 控件的显示和控制。VideoView 控件可以用于视频播放的显示控制。VideoView 类提供了视频播放的开始、暂停等功能。通过 VideoView 类可以播放本地的视频文件,也可以播放网络上的视频文件。

Android 系统支持 MP4 的 H.264、3GP 和 WMV 视频的解析。但是,由于模拟器性能的限制,在模拟器上能正常播放的视频分辨率和码率都比较低。为了达到比较好的效果,能够在 Android 手机上进行调试会更加方便。

下面通过一个实例来实现视频的播放。

【例 10-5】 开场动画的制作片场。

其实现步骤如下:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 Open_animat。

(2) 修改新建项目的 res\layout 目录下的布局文件 main.xml,在代码中添加一个 FrameLayout 帧布局管理器,并在该布局管理器中添加一个 ImageView 控件,用于显示小鸭图像,另外,还需要为添加的帧布局管理器设置背景图片。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:background = "@drawable/bj2">
    < ImageView
        android:id = "@ + id/rabbit"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:src = "@drawable/duck1" />
</FrameLayout >
```

(3) 在 res\layout 目录下创建一个布局文件 start.xml,在文件中添加一个居中显示的线性布局管理器,并在该布局管理器中添加一个 VideoView 控件,用于播放开场动画视频文件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:layout_gravity = "center"
    android:orientation = "vertical" >
    < VideoView
        android:id = "@ + id/video"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />
</LinearLayout >
```

(4) 在 res 目录下新建一个 raw 文件夹,把视频文件 avd 放置于该文件夹中。

(5) 在 src\fs.open_animat 目录下创建一个名为 StartActivity 的活动文件,并重写其 onCreate()方法,在该方法中先获取 VideoView 控件,并获取要播放的文件对应的 URI,接着为 VideoView 控件指定要播放的视频,并让其获得焦点,再调用 start()方法开始播放视频,最后为 VideoView 添加完成事件监听器,在重写的 onCompletion()方法中调用 startMain()方法进入到游戏主界面。代码为:

```
public class StartActivity extends Activity
{
    private VideoView video;                                    //声明 VideoView 对象
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.start);
        video = (VideoView) findViewById(R.id.video);         //获取 VideoView 控件
        //获取要播放的文件对应的 URI
        Uri uri = Uri.parse("android.resource://com.mingrisoft/" + R.raw.avd);
        video.setVideoURI(uri);                               //指定要播放的视频
        video.requestFocus();                                 //VideoView 获取焦点
        try {
            video.start();                                    //开始播放视频
        } catch (Exception e) {
```

```
            e.printStackTrace();                        //输出异常信息
        }
        //为 VideoView 添加完成事件监听器
        video.setOnCompletionListener(new OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                startMain();                            //进入游戏主界面
            }
        });
    }
//编写进入游戏主界面的 startMain()方法,在该方法中创建一个新的 Intent,以启动游戏主界面
//的 Activity
    //进入游戏主界面
    private void startMain() {
        Intent intent = new Intent(StartActivity.this, MainActivity.class);   //创建 Intent
        startActivity(intent);                          //启动新的 Activity
        StartActivity.this.finish();                    //结束当前 Activity
    }
}
```

(6) 打开 MainActivity 文件,重写其 onCreate()方法。代码为:

```
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
   {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

(7) 打开 AndroidMainfest.xml 文件,在文件中配置项目中应用的 Activity。在此首先将主 Activity 设置为 StartActivity,接着配置 MainActivity,主代码为:

```
        <activity
            android:name=".StartActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity"/>
    </application>
</manifest>
```

运行程序,首先播放指定的视频,视频播放完后,将进入如图 10-4 所示的游戏主界面。



图 10-4　游戏主界面

10.4　摄像头的实现

当前的手机设备在硬件上都是支持摄像头的，其拍照功能已成为 Android 手机的基本功能。下面介绍如何具体地操作多媒体应用。

10.4.1　摄像头的拍照功能

Android 系统提供了对摄像头拍照的支持，当然这需要手机本身的硬件支持，Android 中的 Camera 类提供了相关方法。

1. SurfaceView 类的控制

对于 SurfaceView 类的控制，可以使用 SurfaceHolder 接口来完成。首先，需要获取该 SurfaceView 的 SurfaceHolder 接口，使用方法为：

```
SurfaceHolder getHolder()
```

其中，返回值便是控制接口 SurfaceHolder 类。在 SurfaceHolder 类中，可以设置添加回调实现、设置显示的固定大小、获得数据来源等，使用方法分别为：

```
Void addCallback(SurfaceHolder.Callback callback)
Void setFixedSize(int width, int height)
Void setType(int type)
```

2. SurfaceView 图像变化

对于 SurfaceView 的变化监控，需要实现 SurfaceHolder. Callback 接口。在该接口中

针对 SurfView 的创建、变化以及销毁时都可以进行相应的处理。在 SurfaceHolder. Callback 接口中，分别对应以下 3 个方法。

```
Public void surfaceCreated(SurfaceHolder arg0)                                    //创建时
Public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3)      //变化时
Public void surfaceDestroyed(SurfaceHolder arg0)                                  //销毁时
```

下面通过一个实例来实现摄像头的拍照功能。

【例 10-6】 实现 Android 手机摄像头的拍照功能。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li10_6Photograph。

(2) 打开 res\layout 目录下的 main. xml 文件，用于实现摄像头拍照内容的预览位置。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:baselineAligned = "false"
    android:orientation = "horizontal"
    android:background = "#bbbfff" >
    <FrameLayout
        android:id = "@ + id/camera_preview"
        android:layout_width = "0dip"
        android:layout_height = "fill_parent"
        android:layout_weight = "1" />
    <LinearLayout
        android:layout_width = "wrap_content"
        android:layout_height = "fill_parent"
        android:gravity = "center"
        android:orientation = "vertical" >
        <Button
            android:id = "@ + id/button_capture"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center"
            android:layout_marginBottom = "5dip"
            android:text = "拍照" />
        <Button
            android:id = "@ + id/button_camera"
            android:layout_width = "315dp"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center"
            android:text = "返回" />
    </LinearLayout>
</LinearLayout>
```

(3) 打开 src\fs. li10_6photograph 包下的 MainActivity. java 文件，在文件中启用 Activity 类，实现摄像头的拍照功能。代码为：

```
package fs.li10_6photograph;
import java.io.File;
```

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.hardware.Camera;
import android.hardware.Camera.AutoFocusCallback;
import android.hardware.Camera.PictureCallback;
import android.hardware.Camera.ShutterCallback;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.FrameLayout;
public class MainActivity extends Activity {
    public static final int MEDIA_TYPE_IMAGE = 1;          //图片媒体类型
    private Camera mCamera;                                //摄像头类的对象
    private CameraPreview mPreview;                        //SurfaceView 对象
    private Button captureButton;                          //"拍照"按钮
    private Button cameraButton;                           //返回摄像头
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Window window = getWindow();                       //得到窗口
        requestWindowFeature(Window.FEATURE_NO_TITLE);     //没有标题
        window.setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);       //设置全屏
        window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON); //保持屏幕亮
        setContentView(R.layout.main);
        captureButton = (Button) findViewById(R.id.button_capture);
        captureButton.setOnClickListener(new OnClickListener() {         //"拍照"按钮事件
                    @Override
                    public void onClick(View v) {
                        //先启动自动对焦
                        mCamera.autoFocus(new AutoFocusCallback() {
                            @Override
                            public void onAutoFocus(boolean success,
                                    Camera camera) {
                            }
                        });
                        // 1.75 秒后启动拍照
                        Timer timer = new Timer();
                        timer.schedule(new TimerTask() {
                            @Override
                            public void run() {
                                //拍照函数
```

```java
                            mCamera.takePicture(mShutter, null, mPicture);
                        }
                    }, 1750);
                }
            });
        cameraButton = (Button) findViewById(R.id.button_camera);
        cameraButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //"返回"按钮(拍完照片之后需要重新启动 Camera 的 Preview)
                mCamera.startPreview();
            }
        });
        mCamera = getCameraInstance();                    //获取 Camera 对象的实例
        //初始化 SurfaceView
        mPreview = new CameraPreview(this, mCamera);
        FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
        preview.addView(mPreview);                    //将 SurfaceView 添加到 FrameLayout 中
        Camera.Parameters params = mCamera.getParameters();//设置相机的属性
        params.setJpegQuality(100);                       //将 JPEG 质量设置为最好
        //将散光灯模式设置为自动调节
        params.setFlashMode(Camera.Parameters.FLASH_MODE_AUTO);
        mCamera.setParameters(params);
    }
    //PictureCallback 回调函数的实现
    private PictureCallback mPicture = new PictureCallback() {
        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);
            if (pictureFile == null) {
                return;
            }
            //将照片数据 data 写入指定的文件
            try {
                FileOutputStream fos = new FileOutputStream(pictureFile);
                fos.write(data);
                fos.close();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    };
    //快门的回调函数实现
    private ShutterCallback mShutter = new ShutterCallback() {
        @Override
        public void onShutter() {
        }
    };
    //释放 Camera 对象(务必实现)
    private void releaseCamera() {
        if (mCamera != null) {
            mCamera.release();
```

```
                mCamera = null;
            }
        }
        public static Camera getCameraInstance() {
            Camera c = null;
            try {
                c = Camera.open();
            } catch (Exception e) {
                e.printStackTrace();
            }
            return c;
        }
        //在指定路径创建照片文件
        public static File getOutputMediaFile(int type) {
            //指定照片存放的目录,在 SD 根目录下的一个文件夹中
              File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory
    (Environment.DIRECTORY_PICTURES),"CameraUseApp");
            //文件夹不存在,则创建该文件夹
            if (!mediaStorageDir.exists()) {
                if (!mediaStorageDir.mkdirs()) {
                    Log.d("CameraUse", "failed to create directory");
                    return null;
                }
            }
            String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
            //创建照片文件
            File mediaFile;
            if (type == MEDIA_TYPE_IMAGE) {
                mediaFile = new File(mediaStorageDir.getPath() + File.separator
                        + "IMG_" + timeStamp + ".jpg");
            } else {
                return null;
            }
            return mediaFile;
        }
        @Override
        public boolean onTouchEvent(MotionEvent event) {
            //触摸屏幕自动对焦
            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                mCamera.autoFocus(new AutoFocusCallback() {
                    @Override
                    public void onAutoFocus(boolean success, Camera camera) {
                    }
                });
            }
            return super.onTouchEvent(event);
        }
        @Override
        protected void onPause() {
            super.onPause();
            releaseCamera();
        }
    }
```

(4) 在 src\fs. li10_6photograph 包下创建一个 CameraPreview. java 文件，用于实现 SurfaceView，SurfaceView 类用来获取摄像头的实景内容。代码为：

```
package fs.li10_6photograph;
import java.io.IOException;
import android.content.Context;
import android.hardware.Camera;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
public class CameraPreview extends SurfaceView implements
        SurfaceHolder.Callback {
    private SurfaceHolder mHolder;
    private Camera mCamera;
    public CameraPreview(Context context, Camera camera) {
        super(context);
        mCamera = camera;
        mHolder = getHolder();
        mHolder.addCallback(this);
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }
    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        if (holder.getSurface() == null)
            return;
        try {
            mCamera.setPreviewDisplay(holder);
        } catch (IOException e) {
            e.printStackTrace();
        }
        mCamera.startPreview();
    }
    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
        if (mHolder.getSurface() == null)
            return;
        mCamera.stopPreview();
        try {
            mCamera.setPreviewDisplay(mHolder);
        } catch (IOException e) {
            e.printStackTrace();
        }
        mCamera.startPreview();
    }
    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
    }
}
```

(5) 打开 AndroidManifest. xml 文件，为摄像头的拍照添加权限。代码为：

```
…
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
```

```
<! -- 设置摄像头的拍照权限 -- >
    < uses - permission android:name = "android.permission.CAMERA" />
    < uses - permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE" />
    < uses - feature android:name = "android.hardware.camera" />
    < uses - feature android:name = "android.hardware.camera.autofocus" />
…
```

运行程序,效果如图 10-5 所示。



图 10-5　摄像头拍照界面

10.4.2　实现摄像头录制

MediaRecorder 除了可用于录制音频外,还可用于录制视频。使用 MediaRecorder 录制视频和录制音频相似,只是录制视频时不仅需要采集声音,还需要采集图像。为了让 MediaRecorder 录制时采集图像,应该在调用 setAutioSource(int audio_source)方法前调用 setVideoSource(int video_source)方法设置图像来源。

除此之外,还需要在调用 setOutputFormat()设置输出文件格式之后进行以下操作。

(1) 调用 MediaRecorder 对象的 setVideoEncoder()、setVideoEncodingBitRate(int bitRate)、setVideoFrameRate 设置所录制的视频的编码格式、编码位率、每秒多少帧等,这些参数可以控制所录制的视频的品质、文件的大小。一般来说,视频品质越好,视频文件越大。

(2) 调用 MediaRecorder 的 setPreviewDisplay(Surface sv)方法设置使用哪个 SurfaceView 来显示视频预览。

下面通过一个实例来实现摄像头的录制功能。

【例 10-7】 本例为利用 MediaRecorder 实现视频录制功能。界面中提供了两个按钮，分别用于控制开始、结束录制，提供了一个 SurfaceView 用于显示视频预览。

其实现步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 l0_7Camerarecord。

(2) 打开 res\layout 目录下的 main. xml 文件，用于实现摄像头的录制界面。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal">
< LinearLayout
    android:orientation = "horizontal"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:gravity = "center_horizontal">
< ImageButton
    android:id = "@ + id/record"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:src = "@drawable/record"/>
< ImageButton
    android:id = "@ + id/stop"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:src = "@drawable/stop"/>
</LinearLayout >
<! -- 显示视频预览的 SurfaceView -- >
< SurfaceView
    android:id = "@ + id/sView"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"/>
</LinearLayout >
```

(3) 打开 src\fs. li10_7camerarecord 包下的 MainActivity. java 文件，用于实现摄像头的录制功能。代码为：

```
Package fs.li10_7camerarecord;
import java.io.File;
import java.io.IOException;
import java.util.Timer;
import java.util.TimerTask;
import android.media.MediaRecorder;
import android.view.SurfaceView;
public class MainActivity extends Activity implements OnClickListener
{
    //程序中的两个按钮
    ImageButton record , stop;
    //系统的视频文件
    File videoFile ;
```

```
MediaRecorder mRecorder;
//显示视频预览的 SurfaceView
SurfaceView sView;
//记录是否正在进行录制
private boolean isRecording = false;
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    //获取程序界面中的两个按钮
    record = (ImageButton) findViewById(R.id.record);
    stop = (ImageButton) findViewById(R.id.stop);
    //让 stop 按钮不可用
    stop.setEnabled(false);
    //为两个按钮的单击事件绑定监听器
    record.setOnClickListener(this);
    stop.setOnClickListener(this);
    //获取程序界面中的 SurfaceView
    sView = (SurfaceView) this.findViewById(R.id.sView);
    //下面设置 Surface 不需要自己维护缓冲区
    sView.getHolder().setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    //设置分辨率
    sView.getHolder().setFixedSize(320, 280);
    //设置该控件让屏幕不会自动关闭
    sView.getHolder().setKeepScreenOn(true);
}
@Override
public void onClick(View source)
{
    switch (source.getId())
    {
        //单击录制按钮
        case R.id.record:
            if (!Environment.getExternalStorageState().equals(
                android.os.Environment.MEDIA_MOUNTED))
            {
                Toast.makeText(MainActivity.this
                    , "SD 卡不存在,请插入 SD 卡!"
                    , 5000)
                    .show();
                return;
            }
            try
            {
                //创建保存录制视频的视频文件
                videoFile = new File(Environment
                    .getExternalStorageDirectory()
                    .getCanonicalFile() + "/myvideo.mp4");
                //创建 MediaPlayer 对象
                mRecorder = new MediaRecorder();
                mRecorder.reset();
                //设置从麦克风采集声音
                mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
```

```java
                    //设置从摄像头采集图像
        mRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
                    //设置视频文件的输出格式(必须在设置声音编码格式、图像编码格式之前设置)
                    mRecorder.setOutputFormat(MediaRecorder
                        .OutputFormat.MPEG_4);
                    //设置声音编码的格式
                    mRecorder.setAudioEncoder(MediaRecorder
                        .AudioEncoder.DEFAULT);
                    //设置图像编码的格式
                    mRecorder.setVideoEncoder(MediaRecorder
                        .VideoEncoder.MPEG_4_SP);
                    mRecorder.setVideoSize(320, 280);
                    //每秒 4 帧
                    mRecorder.setVideoFrameRate(4);
                    mRecorder.setOutputFile(videoFile.getAbsolutePath());
                    //指定使用 SurfaceView 来预览视频
                    mRecorder.setPreviewDisplay(sView.getHolder().getSurface());
                    mRecorder.prepare();
                    //开始录制
                    mRecorder.start();
                    System.out.println("====recording====");
                    //让 record 按钮不可用
                    record.setEnabled(false);
                    //让 stop 按钮可用
                    stop.setEnabled(true);
                    isRecording = true;
                }
                catch (Exception e)
                {
                    e.printStackTrace();
                }
                break;
            //单击停止按钮
            case R.id.stop:
                //如果正在进行录制
                if (isRecording)
                {
                    //停止录制
                    mRecorder.stop();
                    //释放资源
                    mRecorder.release();
                    mRecorder = null;
                    //让 record 按钮可用
                    record.setEnabled(true);
                    //让 stop 按钮不可用
                    stop.setEnabled(false);
                }
                break;
        }
    }
}
```

(4) 授予录制视频权限。打开 AndroidManifest.xml 文件,添加以下代码授予权限:

```
</application>
    <!-- 授予该程序录制声音的权限 -->
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
     <!-- 授予该程序使用摄像头的权限 -->
     <uses-permission android:name="android.permission.CAMERA"/>
     <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
       <!-- 授予使用外部存储器的权限 -->
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
</manifest>
```

运行程序,效果如图 10-6 所示。



图 10-6　录制视频界面图

10.5 传　感　器

目前手机在人们的日常生活中越来越重要的一个原因就是其具备了各种各样的传感器,主要包括 GPS 位置传感器、方向传感器、加速度传感器、重力传感器、温度传感器、压力传感器、磁场传感器、陀螺仪、亮度传感器和邻近度传感器等,增添了用户位置的获取以及交互方式的改变。

10.5.1 GPS 位置传感器

全球定位系统(Global Positionins System,GPS)最早应用于军事上,现在已经被越来越广泛地应用于人们的日常生活中,例如常见的车辆 GPS 导航仪、智能手机上的 GPS 应用等。GPS 定位是基于卫星的,因此又被称为全球卫星定位系统,用于 GPS 的卫星通常运行在中距离的圆形轨道上,它可以为地球表面的绝大部分地区提供准确的定位、测速和高精度的时间标准。

由于 GPS 的实用性,越来越多的智能手机开始支持它,Android 系统也不例外。GPS 几乎是每个搭建 Android 平台的手机的必备功能。

在 Android 中提供了对应方法用于检查 GPS 的状态。

1. 获取首次定位时间

在 Android 中提供了 getTimeToFirstFix 方法用于获取 GPS 的首次定位时间。getTimeToFirstFix 方法用于获取最新的 GPS 引擎重新启动至收到的首次定位所需的时间,其单位为毫秒(ms)。

注意: *在空旷的地方,GPS 定位时间较短;而在障碍物较多的地方,定位时间较长。*

getTimeToFirstFix 方法的调用格式为:

```
public int getTimeToFirstFix()
```

2. 获取最大卫星数量

在 Android 中提供了 getMaxStatellites 方法用于获取卫星列表中可返回的最大卫星数目。这个数目是 getStatellites 方法能够得到的最大卫星数目，但并不代表当前实际获得的卫星数量。其返回值一般为 255。

getMaxStatellites 方法的调用格式为：

```
public int getMaxStatellites()
```

3. 获取 GPS 卫星状态

在 Android 中提供了 getStatellites 方法用于获取 GPS 引擎的当前状态，该方法返回一个 GpsStatellites 的对象数组值，其中包含了卫星列表。

getStatellites 方法的调用格式为：

```
public Iterable<GpsSatellite> getSatellites()
```

以下实例用于获取 GPS 的卫星状态。

【例 10-8】 实现 GPS 的卫星状态效果。

其实现步骤如下：

(1) 在 Eclipse 中建立一个 Android 应用项目，命名为 GPS_2。

(2) 编写布局文件，用于实现一个按钮和一个文本框布局。打开 res\layout 目录下的 main.xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#F0F0F0F0">
<Button
android:layout_height = "wrap_content"
android:layout_width = "match_parent"
android:id = "@ + id/button1"
android:text = "首次耗时"
android:textColor = "#000000"/>
<TextView
android:layout_height = "wrap_content"
android:layout_width = "wrap_content"
android:id = "@ + id/textView1"
android:textColor = "#000000"/>
</LinearLayout>
```

(3) 编写 Activity 文件，用于获取卫星的首次定位耗时。打开 src\fs.gps_2 包下的 MainActivity.java 文件，代码为：

```
package fs.gps_2;
import java.util.List;
import android.app.Activity;
import android.content.Context;
import android.location.Criteria;
import android.location.GpsSatellite;
```

```
import android.location.GpsStatus;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends Activity
{
    /* 第一次调用活动 */
    LocationManager lm;                              //声明 LocationManager
    LocationListener ll;                             //声明监听器
    private GpsStatus gpsStatus;                     //声明 GpsStatus 变量
    private GpsStatus.Listener statusListener;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bt1 = (Button)findViewById(R.id.button1);   //获取按钮对象
       final TextView tv1 = (TextView)findViewById(R.id.textView1);       //获取文本框对象
        bt1.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                // TODO  自动生成的方法存根
                String str = "当前位置:\n";
                //获取系统服务 lm = (LocationManager)getSystemService(Context.LOCATION_
                //SERVICE);
                statusListener  =  new GpsStatus.Listener()//GPS 状态监听器
                {
                    @Override
                    public void onGpsStatusChanged(int event)
                    {
                        // TODO  自动生成的方法存根
                        gpsStatus = lm.getGpsStatus(null);//获取 GPS 状态
                        switch(event)                //分类说明 GPS 状态事件
                        {
                case GpsStatus.GPS_EVENT_FIRST_FIX:        //第一次定位时间 UTC
                            int utc;
                            utc = gpsStatus.getTimeToFirstFix();
                            String str;
                            str = "完成首次定位,耗时" + utc + "毫秒";
            Toast.makeText(getApplicationContext(), str, Toast.LENGTH_LONG).show();
                            break;
                case GpsStatus.GPS_EVENT_SATELLITE_STATUS://获取卫星信息
                              Toast.makeText(getApplicationContext(), "卫星信息更新",
Toast.LENGTH_LONG).show();
                            break;
                        case GpsStatus.GPS_EVENT_STARTED://GPS 系统启动
```

```
                                Toast.makeText(getApplicationContext(), "GPS 系统启动",
Toast.LENGTH_LONG).show();
                             break;
                        case GpsStatus.GPS_EVENT_STOPPED://GPS 系统停止
                                Toast.makeText(getApplicationContext(), "GPS 系统停止",
Toast.LENGTH_LONG).show();
                             break;
                        default:
                             break;
                        }
                    }
                };
                lm.addGpsStatusListener(statusListener); //添加 GPS 状态监听器
                ll = new LocationListener()              //更新位置
                {
                    @Override
                    public void onLocationChanged(Location location)
                    {
                        // TODO  自动生成的方法存根
                        String str = "当前位置:\n";
                        double latitude;
                        double longitude;
                        latitude = location.getLatitude();//获取纬度
                        longitude = location.getLongitude();//获取经度
                        str = str + "纬度:" + latitude + "\n 经度:" + longitude;
                        tv1.setText(str);
                    }
                    @Override
                    public void onProviderDisabled(String provider)
                    {
                        // TODO  自动生成的方法存根
                          Toast.makeText(getApplicationContext(), "禁用!", Toast.LENGTH_
LONG).show();
                    }
                    @Override
                    public void onProviderEnabled(String provider)
                    {
                        // TODO  自动生成的方法存根
                          Toast.makeText(getApplicationContext(), "使能!", Toast.LENGTH_
LONG).show();
                    }
                    @Override
                    public void onStatusChanged(String provider, int status,
                            Bundle extras)
                    {
                        // TODO  自动生成的方法存根
                     Toast.makeText(getApplicationContext(), "状态改变!", Toast.LENGTH_
LONG).show();
                    }
                };
            lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, ll);
            }
        });
    }
}
```

（4）授予权限。打开 AndroidManifest. xml 文件，添加以下代码：

```
    </application>
    <!-- 授予 GPS 权限 -->
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
</manifest>
```

运行程序，得到模拟器界面，单击界面中的"首次耗时"按钮，然后返回到 DDMS 的 Emulator Control 面板中，改写经/纬度数据，并单击面板中的 send 按钮，即可向 Android 模拟器发送 GPS 定位信息，效果如图 10-7 所示。



图 10-7　获取 GPS 状态

10.5.2　传感器介绍

为了方便传感器的访问，Android 提供了用于访问硬件的 API——android. hardware 包，该包主要提供了用于访问 Camera（相机）和 Sensor（传感器）的类和接口。那么在 Android 中是怎样使用传感器的呢？

在 Android 应用程序中使用传感器依赖于 android. hardware. SensorEventListener 接口，通过该接口可以监听传感器的各种事件。SensorEventListener 接口的代码为：

```
package android.hardware;
public interface SensorEventListener
{
    public abstract void onSensorChanged(SensorEvent event);
    //传感器采样值发生变化时调用
    public abstract void onAccuracyChanged(Sensor sensor, int accuracy);
    //传感器精度发生改变时调用
}
```

接口包括了上段代码中所声明的两个方法，其中，onAccuracyChanged 方法在一般场合中较少用到，常用的方法是 onSensorChanged，它只有一个 SensorEvent 类型的参数 event。

SensorEvent 类代表了一次传感器的响应事件，当系统从传感器获取到信息的变更时，会捕获该信息并向上层返回一个 SensorEvent 类型的对象，这个对象包含了传感器类型(public Sensor sensor)、传感器事件的时间戳(public long timestamp)、传感器数值的精度(public int accuracy)以及传感器的具体数值(public final float[] values)。

其中的 values 值非常重要，其数据类型是 float[]，它代表了从各种传感器采集回的数值信息，该 float 型的数组最多包含 3 个成员，而根据传感器的不同，values 中各成员所代表的含义也不同。例如，通常温度传感器仅仅传回一个用于表示温度的数值，而加速度传感器则需要传回一个包含 X、Y、Z 3 个轴上的加速度数值，同样的一个数据"10"，如果是从温度传感器传回可能代表 10°C，如果是从亮度传感器传回则可能代表数值为 10 的亮度单位，等等。

应用程序可以通过 Sensor 类型和 values 数组的值来正确地处理并使用传感器传回的值。为了正确地理解传感器所传回的数值，这里首先介绍 Android 所定义的两个坐标系，即世界坐标系(world coordinate-system)和旋转坐标系(rotation coordinate-system)。

1. 世界坐标系

如图 10-8 所示，这个坐标系定义了从一个特定的 Android 设备上看待外部世界的方式，主要是以设备的屏幕为基准定义的，并且该坐标系依赖的是屏幕的默认方向，不因为屏幕显示方向的改变而改变。

坐标系以屏幕的中心为圆点，介绍如下。

- X 轴：X 轴的方向是沿着屏幕的水平方向从左向右，如果手机不是正方形的，较短的边需要水平放置，较长的边需要垂直放置。
- Y 轴：Y 轴的方向是从屏幕的左下角开始沿着屏幕的垂直方向指向屏幕的顶端。
- Z 轴：将手机放在桌子上，Z 轴的方向是从手机指向天空。

2. 旋转坐标系

如图 10-9 所示，其中的球体可以理解为地球，这个坐标系是专用于方位传感器(Orientation Sensor)的，其可以理解为一个"反向的(inverted)"世界坐标系，方位传感器即用于描述设备所朝向的方向传感器，而 Android 为描述这个方向定义了一个坐标系，这个坐标系也由 X、Y、Z 轴构成，特别之处是方向传感器所传回的数值是屏幕从标准位置(屏幕水平朝上且位于正北方向)开始分别以 3 个坐标轴所旋转的角度。使用方位传感器的典型用例即"电子罗盘"。

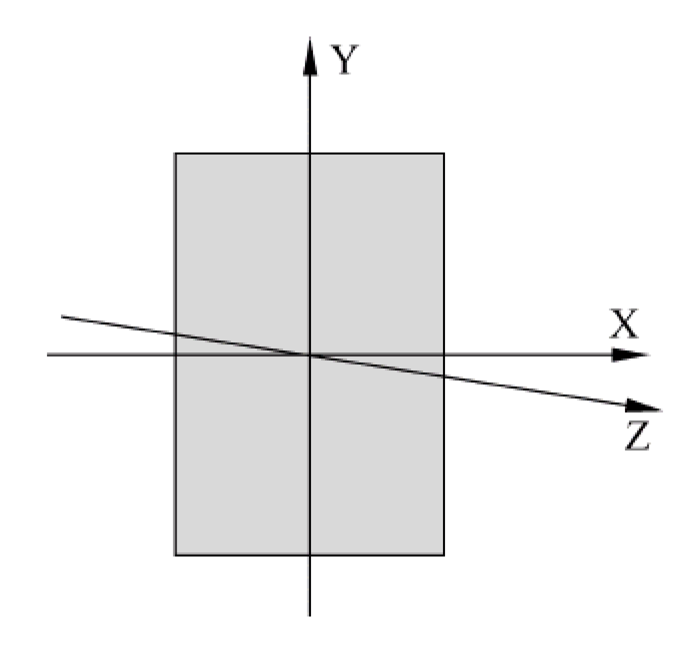


图 10-8 Android 设备的世界坐标系

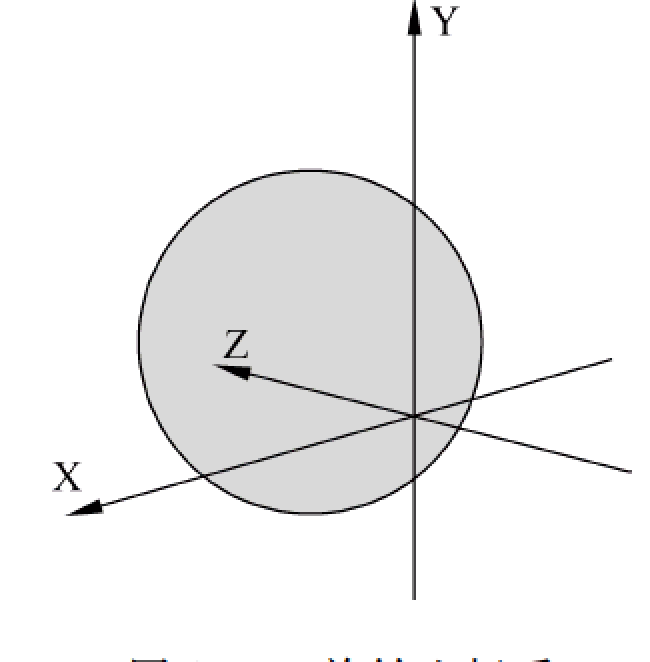


图 10-9 旋转坐标系

对该坐标系中的X、Y、Z轴分别介绍如下。

- X轴：即Y轴与Z轴的向量积Y.Z，方位是与地球面相切并且指向地理位置的西方。
- Y轴：设备当前所在位置与地面相切并且指向地磁北极的方向。
- Z轴：设备所在位置指向地心的方向，因此这里进一步介绍访问传感器所传回的values[]数组中各个数值所表示的含义，作为对values[]值的一个实例说明。当方向传感器感应到方位变化时会返回一个包含变化结果数值的数组，即values[]，数组的长度为3，它们分别代表的含义如下。
- values[0]：该值表示方位，也就是手机绕着Z轴旋转的角度。0表示北(North)；90表示东(East)；180表示南(South)；270表示西(West)。如果values[0]的值正好是这4个值，并且手机是水平放置，表示手机的正前方就是这4个方向。可以利用这个特性来实现电子罗盘。
- values[1]：该值表示倾斜度或手机翘起的程度。当手机绕着X轴倾斜时该值发生变化，values[1]的取值范围是−180≤values[1]≤180。

假设将手机屏幕朝上水平放在桌子上，如果这时桌子是完全水平的，values[1]的值应该是0(由于很少有桌子是绝对水平的，因此该值很可能不为0，但一般都是−5和5之间的某个值)。这时从手机顶部开始抬起，直到将手机沿X轴旋转180度(屏幕向下水平放在桌面上)。在这个旋转过程中，values[1]会在0～180变化，也就是说，从手机顶部抬起时，values[1]的值会逐渐变小，直到等于−180。如果从手机底部开始抬起，直到将手机沿X轴旋转180°，这时values[1]会在0～180变化。也就是values[1]的值会逐渐增大，直到等于180。可以利用values[1]和下面要介绍的values[2]来测量桌子等物体的倾斜度。

- values[2]：表示手机沿着Y轴的滚动角度，其取值范围是−90≤values[2]≤90。假设将手机屏幕朝上水平放在桌面上，如果这时桌面是平的，values[2]的值应为0。将手机左侧逐渐抬起时，values[2]的值逐渐变小，直到手机垂直于桌面放置，这时values[2]的值是−90。将手机右侧逐渐抬起时，values[2]的值逐渐增大，直到手机垂直于桌面放置，这时values[2]的值是90。在垂直位置时继续向右或向左滚动，values[2]的值会继续在−90～90变化。

3. 传感器战例

下面通过一个实例来实现传感器的获取。

【例10-9】 在模拟器中实现传感器的获取。

其实现步骤如下：

(1) 在Eclipse中创建一个Android应用项目，命名为li10_9GetSensor。

(2) 打开res\layout目录下的main.xml文件，在文件中声明一个Button控件和3个TextView控件。代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
xmlns:tools = "http://schemas.android.com/tools"
android:layout_width = "match_parent"
android:layout_height = "match_parent"
tools:context = ".SensorTest"
android:orientation = "vertical"
```

```
android:background = "#bbbfff">
<Button
  android:id = "@ + id/button"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:text = "获取传感器" />
<TextView
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:id = "@ + id/v"
android:textSize = "30px"/>
<TextView
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:id = "@ + id/vx"
android:textSize = "50px"/>
<TextView
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:id = "@ + id/vy"
android:textSize = "50px"/>
<TextView
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:id = "@ + id/vz"
android:textSize = "50px"/>
</LinearLayout>
```

(3) 打开 src\fs.li10_9getsensor 包下的 MainActivity.java 文件，用于实现传感器的获取。代码为：

```
package fs.li10_9getsensor;
import java.util.List;
import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
public class MainActivity extends Activity implements SensorEventListener{
    private SensorManager sensorManager = null;
    private Sensor gyroSensor = null;
    private TextView vX;
    private TextView vY;
    private TextView vZ;
    private TextView v;
    private Button button;
    private static final float NS2S = 1.0f / 1000000000.0f;
```

```
private float timestamp;
private float[] angle = new float[3];
@SuppressWarnings("deprecation")
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    vX = (TextView) findViewById(R.id.vx);
    vY = (TextView)findViewById(R.id.vy);
    vZ = (TextView)findViewById(R.id.vz);
    v = (TextView)findViewById(R.id.v);
    button = (Button)findViewById(R.id.button);
    sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    gyroSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
    button.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View arg0) {
            // TODO  自动生成的方法存根
            //声明可变字符串
            StringBuffer sb = new StringBuffer();
            //获取手机全部的传感器
            List<Sensor> sensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
            //迭代输出获得的传感器
            for (Sensor sensor : sensors) {
                sb.append(sensor.getName().toString());
                sb.append("\n");
                Log.i("Sensor", sensor.getName().toString());
                }
            //给文本控件赋值
            v.setText(sb.toString());
            }
        });
    }
public MainActivity() {
    // TODO  自动生成的方法存根
    angle[0] = 0;
    angle[1] = 0;
    angle[2] = 0;
    timestamp = 0;
    }
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
    }
@Override
protected void onPause() {
    // TODO 自动生成的方法存根
    super.onPause();
    sensorManager.unregisterListener(this);          //解除监听器注册
    }
```

```
    @Override
    protected void onResume() {
        // TODO  自动生成的方法存根
        super.onResume();
          sensorManager. registerListener ( this,  gyroSensor, SensorManager. SENSOR _ DELAY _
NORMAL);                                                        //为传感器注册监听器
        }
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // TODO  自动生成的方法存根
        }
    @Override
    public void onSensorChanged(SensorEvent event) {
        //方向传感器提供 3 个数据,分别为 azimuth、pitch 和 roll
        // azimuth: 方位,返回水平时磁北极和 Y 轴的夹角,范围为 0°～360°
        // 0° = 北,90° = 东,180° = 南,270° = 西
        // pitch: X 轴和水平面的夹角,范围为 - 180°～180°
        //当 Z 轴向 Y 轴转动时,角度为正值
        // roll: Y 轴和水平面的夹角,由于历史原因,范围为 - 90°～90°
        //当 X 轴向 Z 轴移动时,角度为正值
        vX.setText("Orientation X: " + event.values[0]);
        vY.setText("Orientation Y: " + event.values[1]);
        vZ.setText("Orientation Z: " + event.values[2]);
        }
}
```

运行程序,效果如图 10-10(a)所示,单击屏幕中的“获取传感器”按钮,效果如图 10-10(b)所示。



(a) 获取传感器界面　　(b) 显示传感器类型

图 10-10　传感器获取

Android 中的很多游戏都使用了重力传感器技术，下面简单介绍一下重力传感器。以屏幕的左下方为原点，箭头指向的方向为正，从 $-10 \sim 10$，以浮点数为等级单位，想象如图 10-11 所示的情形。

图 10-11　重力传感器坐标图

其数值显示为：

- 手机屏幕向上（Z 轴朝天）水平放置的时候，x、y、z 的值分别为 0、0、10；
- 手机屏幕向下（Z 轴朝地）水平放置的时候，x、y、z 的值分别为 0、0、-10；
- 手机屏幕向左侧放（X 轴朝天）的时候，x、y、z 的值分别为 10、0、0；
- 手机竖直（Y 轴朝天）向上的时候，x、y、z 的值分别为 0、10、0；
- 其他依类推，规律就是"朝天的是正数，朝地的是负数"。

因此，利用 x、y、z 3 个值求三角函数，就可以精确地检测手机的运动状态了。

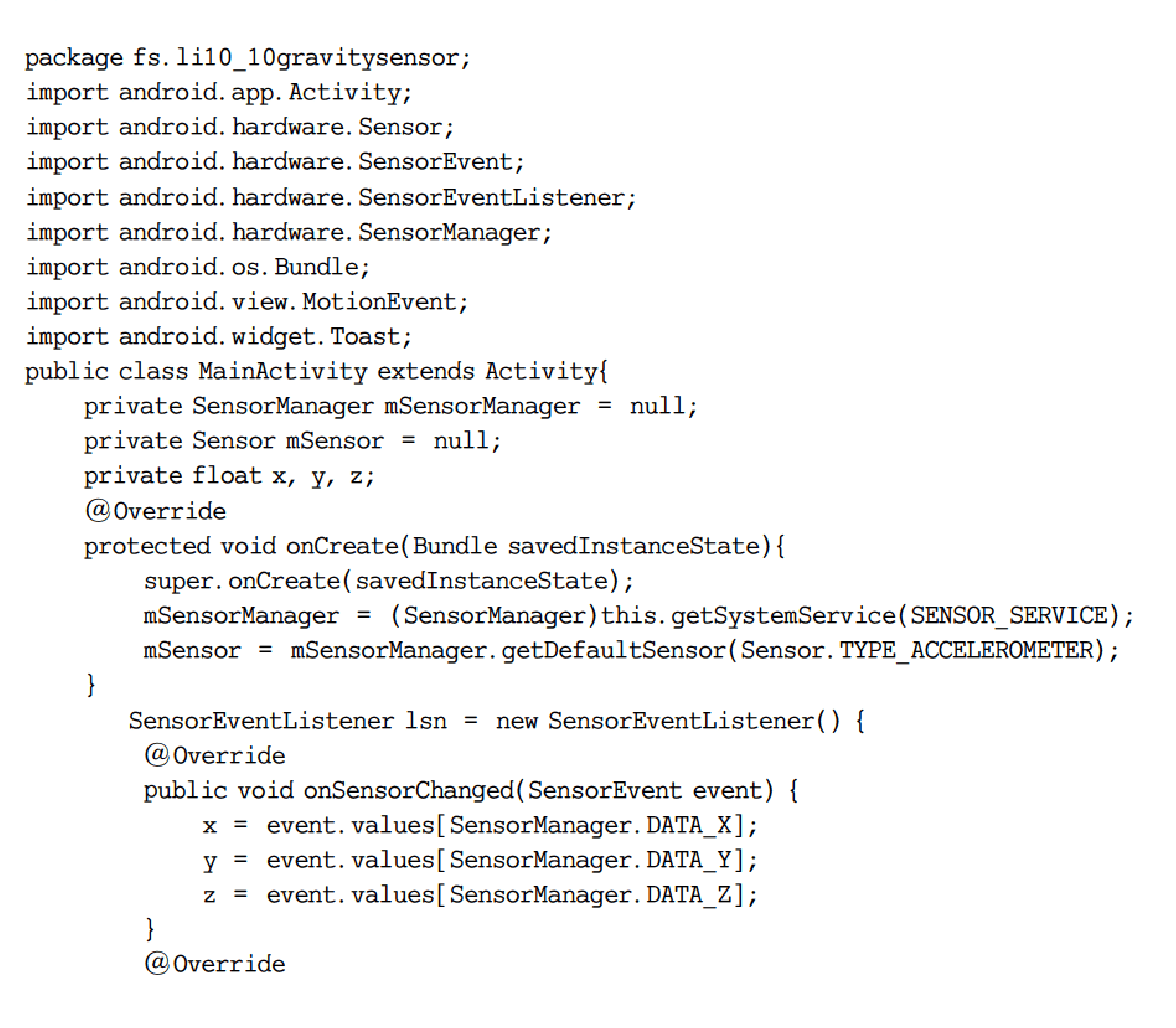
下面通过一个实例来演示重力传感器。

【例 10-10】 测试重力传感器。

其实现步骤如下：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li10_10GravitySensor。

（2）打开 src\fs.li10_10gravitysensor 包下的 MainActivity.java 文件，实现重力传感器的测试。代码为：



```
package fs.li10_10gravitysensor;
import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.MotionEvent;
import android.widget.Toast;
public class MainActivity extends Activity{
    private SensorManager mSensorManager = null;
    private Sensor mSensor = null;
    private float x, y, z;
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        mSensorManager = (SensorManager)this.getSystemService(SENSOR_SERVICE);
        mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }
       SensorEventListener lsn = new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            x = event.values[SensorManager.DATA_X];
            y = event.values[SensorManager.DATA_Y];
            z = event.values[SensorManager.DATA_Z];
        }
        @Override
```

```
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
            // TODO  自动生成的方法存根
        }
    };
    @Override
    public boolean onTouchEvent(MotionEvent event){
        if(event.getAction() == MotionEvent.ACTION_DOWN){
              mSensorManager.registerListener(lsn, mSensor, SensorManager.SENSOR_DELAY_
GAME);
            String str = "x=" + x + "; y=" + y + "; z=" + z;
            Toast.makeText(getApplicationContext(), str, Toast.LENGTH_LONG).show();
        }
        return super.onTouchEvent(event);
    }
    @Override
    public void onResume(){
        mSensorManager.registerListener(lsn, mSensor, SensorManager.SENSOR_DELAY_GAME);
        super.onResume();
    }
    @Override
    public void onPause(){
        mSensorManager.unregisterListener(lsn);
        super.onPause();
    }
}
```

(3) 打开 AndroidManifest.xml 文件,设置传感器权限。代码为:

```
…
    </application>
<!-- 添加传感器权限 -->
<uses-permission android:name="android.hardware.sensor.accelerometer"/>
</manifest>
```

运行程序,单击屏幕上的任意一处,效果如图 10-12 所示。



图 10-12　重力传感器数据

提示:这是在模拟器上截图,由于模拟器无法感应重力,所以显示 x、y、z 的数据都为 0.0。

网上参考资源

http://baike. baidu. com/subview/1241829/9322617. htm? fr=aladdin

http://blog. csdn. net/ithomer/article/details/6741092

http://www. jb51. net/article/37710. htm

http://blog. csdn. net/benweizhu/article/details/7327058

http://www. drovik. com/android/demo/RecorderTrack. rar

http://blog. csdn. net/hellogv/article/details/6406732

http://www. jb51. net/article/40880. htm

http://www. jb51. net/article/46550. htm

http://www. kankanews. com/ICkengine/archives/111395. shtml

http://blog. csdn. net/wangjia55/article/details/7492691

http://blog. chinaunix. net/uid-25422700-id-368672. html

http://blog. csdn. net/sxsj333/article/details/6368246

http://liangruijun. blog. 51cto. com/3061169/647456/

http://www. android-study. com/wangluobiancheng/249. html

http://baike. baidu. com/link? url = bxBdG _ 2zJc2KlKFXAtdHcDOUilyHNnOznK8QFMY1VG3qHqeTHOZAsAqiLtZFmfaC

http://wenku. baidu. com/link? url = CpJjohMaPTDod7aFiiCJYPlxuY1wqYQZF-Nu-ySDbz3pbx76A-q9kF0Il5hFwrycsZsjlnoFXn-U_tSSJgl3CIg4s3pQ0Jjvd7Vbi32d4_q

http://liangruijun. blog. 51cto. com/3061169/657502/

http://vvsongsunny. iteye. com/blog/990698

http://www. oschina. net/question/234345_44374

http://www. cnblogs. com/snake-hand/archive/2012/01/23/2454407. html

http://www. 2cto. com/kf/201304/201488. html

http://www. 2cto. com/kf/201205/131768. html

http://blog. csdn. net/flyfight88/article/details/8602162

http://blog. csdn. net/yuzhongchun/article/details/8956256

http://blog. csdn. net/liuyiming_/article/details/7704923

http://mobile. 51cto. com/aprogram-395278. htm

http://www. cnblogs. com/linjiqin/archive/2011/02/21/1960214. html

http://www. cnblogs. com/over140/archive/2010/11/18/1880391. html

http://blog. 163. com/fan_jianglong@126/blog/static/56170536201242555419455/

http://android. tgbus. com/Android/tutorial/201107/360548. shtml

http://www. cnblogs. com/linjiqin/archive/2011/02/24/1963582. html

http://www. iteye. com/topic/540423

http://www.cnblogs.com/linjiqin/archive/2011/02/22/1961050.html
http://www.cnblogs.com/linjiqin/archive/2011/02/21/1960107.html
http://www.cnblogs.com/classic/archive/2011/06/20/2085055.html
http://blog.csdn.net/liuhaoyutz/article/details/9625153
http://www.cnblogs.com/yyyyy5101/archive/2011/06/20/2085407.html
http://www.oschina.net/question/54100_32483
http://blog.csdn.net/pku_android/article/details/7343258
http://blog.csdn.net/telencool/article/details/7360766
http://www.cnblogs.com/devinzhang/archive/2012/01/18/2325887.html
http://blog.e23.cn/?uid-2027514-action-viewspace-itemid-832367
http://wenku.baidu.com/link?url=DaDaNBV0FdbpvWiixKWzkWNsEhk4vvOFotqMjCjMQRIiB1HdOwr7hVUN00B1XCQu46P297fS9v60yV97h-E85difGuHyxRGHX6fLrWFcVRO
http://zwnjava.iteye.com/blog/1746505
http://www.cnblogs.com/rollenholt/archive/2012/05/17/2506365.html
http://www.cnblogs.com/linjiqin/archive/2011/02/21/1960145.html
http://imshare.iteye.com/blog/770950
http://www.cnblogs.com/linjiqin/archive/2011/02/21/1960185.html
http://www.cnblogs.com/linjiqin/archive/2011/02/23/1962535.html
http://www.cnblogs.com/salam/archive/2010/10/06/1844660.html
http://www.cnblogs.com/mandroid/archive/2011/02/24/1963357.html

参 考 文 献

[1] Wei-Meng Lee. Android 编程入门经典. 何晨光,李洪刚,等译. 北京：清华大学出版社,2012.

[2] 扶松柏. Android 开发从入门到精通. 北京：北京希望电子出版社,兵器工业出版社,2012.

[3] 欧阳零. Android 编程兵书. 北京：电子工业出版社,2014.

[4] 李佐彬. Android 开发入门与实战体验. 北京：机械工业出版社,2011.

[5] 杨明羽. Android 语法范例参考大全. 北京：电子工业出版社,2012.

[6] 李刚. 疯狂 Android 讲义. 北京：电子工业出版社,2011.

[7] 楚无咎. Android 编程经典 200 例. 北京：电子工业出版社,2013.

[8] 吴亚峰,索依娜. Android 核心技术与实例详解. 北京：电子工业出版社,2010.

[9] 明日科技. Android 从入门到精通. 北京：清华大学出版社,2012.

[10] 赵启明. Android 典型技术模块开发详解. 北京：中国铁道出版社,2011.